Mitigating sandwich attacks in Kyber DMM

Arif Akif Yüksel , Oğuzhan Ersoy , Zekeriya Erkin
Cyber Security Group
Department of Intelligent Systems
Delft University of Technology

Abstract

Kyber is a Decentralized Finance (DeFi) system which runs on the Ethereum blockchain. DeFi aims to remove centralized intermediaries such as Market Makers. An Automated Market Maker (AMM), implemented in a smart contract, is a decentralized version of these. Kyber's Dynamic Market Maker (DMM) is a next-generation AMM which solves two issues: Capital Inefficiency (CI) and Impermanent Loss (IL). CI is decreased by an amplification factor which a Liquidity Provider sets upon creaton of a liquidity pool, whereas IL is decreased by dynamic fees. A DMM features two reserves: one real reserve that reflects the true amounts of the two tokens in the pool and one virtual reserve that reflects the amounts after the amplification factor is applied. The vulnerability to a sandwich attack exists because the virtual reserve ratio can be unbalanced by an attacker. This results in slippage for the victim when their transaction gets executed. Finally, the attacker can perform a swap using the incorrect ratio. The research question of this paper is: How can one mitigate sandwich attacks in Kyber DMM? Kyber's current mitigation features slippage protection to protect users from sandwich attacks. The slippage protection is implemented by adding two parameters to the function used when adding liquidity: one for specifying a lower bound for the virtual reserve ratio and one for specifying the upper bound. However, this mitigation is only present in the router. Therefore, users interacting with the pool contract directly remain vulnerable. To show that this is true, we modify Kyber's test case for sandwich attacks to encompass the mint function in the pool contract. The existing mitigation can be broadened by implementing a code correction in the mint function like the one present in the function used when adding liquidity.

1 Introduction

Over the last few years, the popularity of Decentralized Finance (DeFi) has grown exponentially, with the total value locked (TVL) in DeFi being over $48 billion as of June 2021 [1]. With such widespread adoption, a security vulnerability can have severe implications. Such vulnerabilities as well as existing countermeasures should be well documented to help prevent exploitation by bad actors.

The DeFi system of interest for this paper is one known as Kyber [2]. Five security audits have been published for Kyber since it was released in 2017. These audits have brought several security vulnerabilities to light, most of which were subsequently mitigated by the programming team at Kyber [3] [4] [5] [6] [7]. Had the Kyber protocol not been openly auditable, these vulnerabilities would likely be exploited by malicious users, leading to security breaches and potential losses of funds. This shows how important it is for vulnerabilities to be documented.

Kyber's Dynamic Market Maker (DMM) is a new system which is in the beta phase as of writing this paper. The DMM is based on Uniswap's Automated Market Maker (AMM). This works answers the following research question: "How can one mitigate sandwich attacks in Kyber DMM?". There is a work that focuses on sandwich attacks [8] and these papers discuss sandwich attacks [9] [10] [11]. These papers include cryptography-based solutions as well as specific settings a protocol can implement. However, these mitigations go against two of the premises of DeFi systems: transparency and decentralization. Mitigations in literature are discussed more in-depth in Section 4.1.

Sandwich attacks in Kyber's DMM are slightly different from the conventional kind, as the issue is related to the virtual reserves which exist in addition to the real reserves. The virtual reserves are a consequence of the amplification factor, which makes the programmable pricing curve feature possible. There is a mitigation in place for the vulnerability. Namely, the router features slippage protection. This means that if the virtual reserve gets unbalanced by an attacker, the victim's transaction gets reverted. This prevents loss of funds for the victim (except for gas fees). However, the overall vulnerability is not mitigated satisfactorily. Namely, users who interact with the pool contract are still vulnerable to sandwich attacks. This is because the mint function in the pool contract does not feature slippage protection.

The goal of this paper is to show that the vulnerability is indeed present in the mint function and to suggest applying the mitigation to the vulnerable part of the system as well.

It is shown by means of a test that users interacting with the pool contract directly are vulnerable to sandwich attacks. This test is a modified version of the existing one that was made for testing the sandwich attack in the router. For more details, refer to Sections 4.2 and 4.3. We also suggest Kyber to feature slippage protection when minting liquidity too so that the vulnerability to sandwich attacks is mitigated for users who add liquidity through this method as well. Along with this suggestion, a compilation of existing ideas for mitigations against sandwich attacks in literature as well as Kyber's current mitigation are presented. This paper also gives an overview of Kyber, which does not have any papers to its name as far as has been researched.

The structure of this paper is as follows: first, a background is given on (De)centralized Finance, on Kyber and on Automated Market Making. Subsequently, in Section 3, the methodology of this research is explained, and the problem is formally described. Next, possible mitigations are discussed in Section 4. These include mitigations against sandwich attacks in general, Kyber's current mitigation and a suggestion to apply the current mitigation in the pool contract as well. The ethics and reproducibility of this research can be found in Section 5. This Section is followed by the discussion and limitations in Section 6. Finally, conclusions and recommendations for future work are presented in Section 7.

2 Background

Decentralized Finance, despite being novel, has quite an extensive background. In this Section, the reader can find a summary on what DeFi is, why it has emerged and how it compares to Centralized Finance (CeFi) (i.e., the world of mainstream finance as it is now). Furthermore, information on Kyber and its transition to Kyber 3.0 can be found in this section. Finally, a background on Automated Market Making is given. Here, Kyber DMM is introduced and its improvements over the standard constant product-based AMM is explained.

2.1 On (De)centralized Finance

Traditional financial systems are centralized because they contain intermediaries, for example banks and exchanges. These intermediaries allow for interaction between market participants who provide money, such as investors and lenders, and participants who require money, such as entrepreneurs and borrowers. Banks and exchanges are hereby viewed as a central point for interactions. These central points require trust from all parties involved in a financial operation. People have faced many financial crises that have come up over the years with CeFi, like the one in 2008. Even though (government) regulation is enacted in times of failure, these interventions often do not solve the failures entirely. The concept of DeFi has sprung up from this weakness [12].

Bitcoin [13] jumpstarted the cryptocurrency world in 2008, which DeFi is a part of. Bitcoin - particularly, its scripting language, called Script [14] - was not programmable enough for DeFi systems as they are today. In fact, Script is not a Turing complete language, as it has no loops or conditionals. Eventually, Ethereum [15] was created along with the programming language Solidity [16]. Solidity is similar in syntax to Javascript and has taken inspiration from C++ and Python. Furthermore, it is specifically designed to run in the Ethereum Virtual Machine (EVM). As opposed to Bitcoin's scripting language, Solidity is Turing complete and allows for the creation of smart contracts. This in turn allowed for the creation of a plethora of protocols, which Kyber is one of.

Smart contracts are autonomous, self-sufficient, and decentralized programs that handle transactions between two parties without the need for a centralized intermediary [17]. They are autonomous in the sense that they can operate on their own after deployment, so the contracts do not need maintenance. Smart contracts require resources for each operation. Gas fees are used to allocate these resources from the EVM [18]. The fees are collected with each interaction between an agent and the smart contract. This allows for the execution of operations in a decentralized way and is what makes a smart contract self-sufficient. Finally, smart contracts are decentralized programs as they do not have a single host but are rather distributed across an entire network of nodes.

Decentralized Finance systems are peer-to-peer, permissionless financial systems [19]. These systems aim to remove centralized intermediaries in transactions and are thus a direct counterpart to CeFi. Transactions within such systems are processed using smart contracts. A textbook DeFi system has the following properties [10]:

1. DeFi provides non-custodial financial services, meaning every user has full control over their finances.
2. DeFi has a permissionless nature, meaning users cannot be censored or blocked from using the provided services.
3. The states of the protocols in DeFi systems are openly auditable.
4. New financial services and products can be made within DeFi systems by building upon existing protocols.

As opposed to CeFi, there are no intermediaries requiring trust when processing transactions between two parties in DeFi. Instead, transactions in DeFi systems are done by smart contracts, which are characterised as trustless. Smart contracts thereby act as the intermediaries in CeFi, such as brokers or banks. Because they are distributed across a network of nodes, trust is also distributed. Furthermore, smart contracts are easily traceable and verifiable.

There is also the notion of privacy and security. In DeFi, all (pending) transactions are publicly visible in detail on the blockchain, but one cannot infer the identity of a person by their address. However, since a transaction's details are visible in their entirety and executing a transaction takes time, there exists a window for attacks. The sandwich attack is one such attack and is described in section 3.2.

2.2 On Kyber

Kyber is an on-chain liquidity protocol, which means all operations are done on the blockchain. It is the most well-known protocol that uses smart contract-based reserve

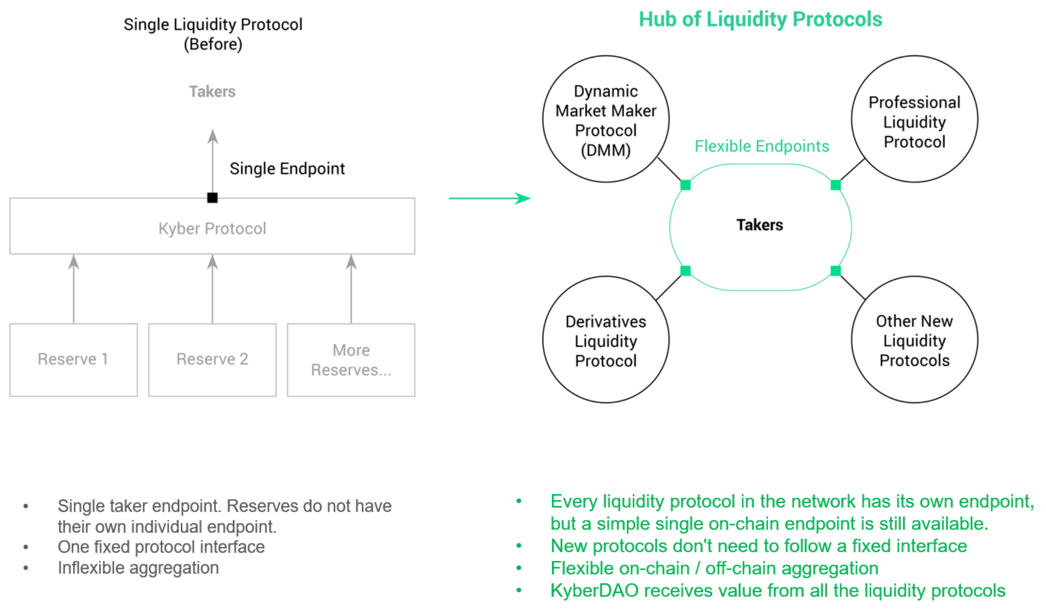
Single Liquidity Protocol
(Before)
Takers
Single Endpoint
Kyber Protocol
Reserve 1
Reserve 2
More Reserves...
Hub of Liquidity Protocols
Dynamic Market Maker Protocol (DMM)
Professional Liquidity Protocol
Flexible Endpoints
Takers
Derivatives Liquidity Protocol
Other New Liquidity Protocols
• Single taker endpoint. Reserves do not have their own individual endpoint.
• One fixed protocol interface
• Inflexible aggregation
• Every liquidity protocol in the network has its own endpoint, but a simple single on-chain endpoint is still available.
• New protocols don't need to follow a fixed interface
• Flexible on-chain / off-chain aggregation
• KyberDAO receives value from all the liquidity protocols

Figure 1: An overview of the architectural changes as a result of the transition to Kyber 3.0 [20]

aggregation for pricing coins [21]. This means that prices are determined by liquidity providers rather than within the smart contract. However, Kyber does not rank very high in terms of trading volume. Other exchanges, such as Uniswap, 1inch and Curve Finance, rank much higher than Kyber in this regard [22].

Kyber is in a transitioning state to Kyber 3.0 as of writing this paper. They deemed this transition necessary to ”remove constraints and enable innovation” [20]. While the existing architecture has a secure foundation and worked well in the beginning, Kyber could not pick up the trends in DeFi in a fast manner because of its restrictive architecture. This is why they are making major architectural changes with this transition. As shown in Figure 1, they are transitioning from a protocol with a single endpoint to a hub of protocols with multiple endpoints. Of these protocols, the DMM protocol [23] is the one of interest for this paper.

Kyber Network is governed by Kyber Network Crystal (KNC) holders. Governance is handled through Kyber Decentralized Autonomous Organization (DAO). On the KyberDAO, KNC holders can stake their KNC and vote for changes that are proposed on the platform. The following is a great example of governance in the KyberDAO: as part of the transition to Kyber 3.0, Kyber Network proposed to upgrade their KNC token (now the KNC Legacy token, or KNCL for short) to the KNC v2 token. Only after enough users voted in favor of this change by staking their KNC tokens on the KyberDAO, the change was realized. This ensures a decentralized way of enacting significant changes to the protocol.

2.3 On the Automated Market Maker

In DeFi, traditional market makers cannot be used as they are in centralized financial systems. This would result in centralization, as the market would be made by a group of individuals whom users would need to trust. The use of order books can be done in a decentralized manner, but this way of determining price is too costly and slow on the blockchain [21]. Therefore, the AMM was introduced [11]. With the use of smart contracts, market making can be done efficiently with the implementation of AMMs. In the schematic depicted in Figure 2, the different kinds of actors as well as their possible user actions are shown. Uniswap's AMM was the first of its kind and it is based on a constant product function:

$$x \times y = k \qquad (1)$$

where x and y are the reserves of the two assets in the pool and k is the constant which must always be equal to the product of x and y [24].

In April 2021, Kyber brought out an improvement over AMM technology and named it the Dynamic Market Maker. Kyber's DMM is a project forked from Uniswap's repository [25]. The DMM solves two key issues in standard Automated Market Maker protocols like the one that is used in Uniswap, which is currently the most popular Decentralized exchange (DEX). These issues are Impermanent Loss (IL) and Capital Inefficiency (CI).

IL refers to temporary losses to Liquidity providers (LPs) due to volatility in the coin for which they provide liquidity. If the LP withdraws coins while the price is different from the initial price at which they added liquidity to their pool, the

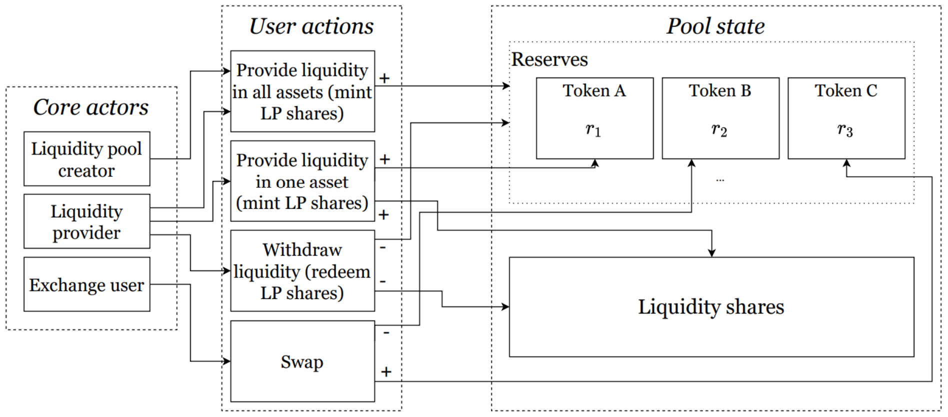

Figure 2: A schematic of the Automated market maker [11]

losses become permanent. LPs make profit if the fees users of the liquidity pool pay for their blocks are higher than their IL. IL is mitigated by the dynamic fees, an improvement over the fixed fees in standard AMMs. Fees are lowered in times of low volatility as an incentive for trading, whereas fees are increased in times of high volatility to increase the profits gained for the LP. Consequently, the negative effects for LPs when high price fluctuations occur in tokens are reduced.

CI is caused by slippage, which is the difference between the price of an asset when the transaction is placed and the price when the transaction is executed. This is where the amplification factor comes into play. This feature changes the constant product formula as mentioned earlier into the following:

$$\alpha x \times \alpha y = \alpha^2 k \tag{2}$$

where α is the amplification factor and the other terms are the same as described before [26]. This equation is maintained in the virtual reserve. Thanks to the amplification factor feature of the DMM, slippage is greatly reduced in a user's transactions. This results in significantly improved capital efficiency compared to a standard AMM. The DMM is in the beta phase as of this moment, but it has already been launched on the mainnet. It is currently in use with real money, with the total value locked in the protocol being over $12M as of June 2021 [27].

3 Sandwich attacks in Kyber DMM

This section first explains the methodology briefly. Here, the reader can find the steps taken in tackling the problem. Next, an introduction of sandwich attacks in general is given. Finally, an explanation on how the DMM is vulnerable to it as well as an outline of the attack steps follow.

3.1 Methodology

This work has mainly been a literature study. The literature study was done through the use of appropriate resources (i.e., scientific papers, whitepapers, Kyber's blog posts and security audits). These resources were found by using search queries in the proper search engines as taught in the Information Literacy 2 course. Examples of such search engines include IEEE Xplore and Google Scholar. Resources were also found by means of collaboration with other group members. These resources were mainly related to the main topic that is DeFi systems.

For the implementation part, the DMM repository was forked and edited locally in Visual Studio Code. The edited version is available on GitHub [28]. An existing test for the sandwich attack was edited in JavaScript to show that the pool contract is indeed vulnerable to sandwich attacks. In this phase, the test failed as expected. Afterwards, a suggestion to implement the code correction in the pool contract in addition to the router was formalized and presented.

3.2 The Sandwich attack

The sandwich attack is based on two transaction reordering techniques known as frontrunning and backrunning [8]. Frontrunning a transaction is possible if the attacker pays more gas fees than the victim, whereas backrunning is possible when they pay less gas fees than the victim. As miners are incentivised to order transactions in descending order of gas fees, this ensures that the techniques succeed. Sandwich attacks have been plaguing the DeFi space since its inception and are a relatively common type of attack in major exchanges such as Uniswap, Bancor and Sushiswap [9].

For an explanation on how a sandwich attack works, consider the following scenario: take a pool with tokens X and Y. A user initiates a transaction in which they want to swap some amount of X for however many of token Y that is worth in the pool. An attacker sees the pending transaction in the mempool and initiates the sandwich attack. They create two transactions and schedule them to execute before and after the victim's transaction by the aforementioned techniques respectively. The first transaction is to swap some amount of X for Y, whereas the second is to swap some amount of Y for X. In a traditional sandwich attack, the first transaction aims to induce extra slippage on the victim's transaction. In particular, the victim gets less of token Y than they expect because of the attacker's first transaction. After this transaction gets executed, they let the victim's

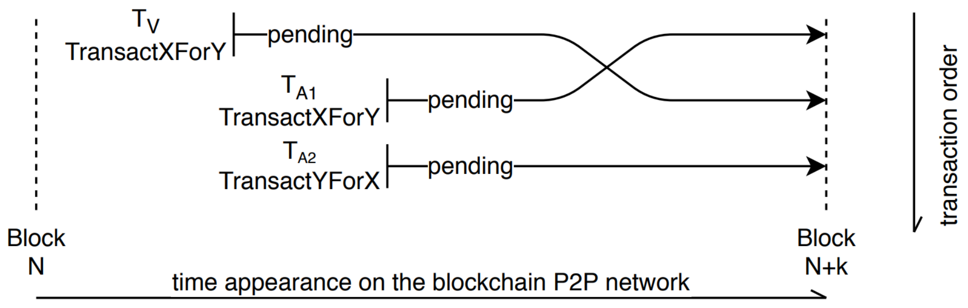

Figure 3: An overview of the sandwich attack [8]

transaction happen. Finally, the second attacker transaction gets executed. This results in the attacker effectively stealing the victim's funds. Thus, the sandwich attack has been completed. See Figure 3 for an overview of the attack as described here.

In the fifth published security audit for Kyber [7], a vulnerability to sandwich attacks was found in the DMM. This attack is possible because of the programmable pricing curve feature, which works by amplifying the assets of a pool by a factor defined by the liquidity provider during creation. This feature results in the pool having two types of reserves, one real and one virtual. The problem the DMM is facing is the possibility for an attacker to unbalance the virtual reserve ratio between the two assets in a liquidity pool. Currently, the vulnerability is present for users interacting directly with the pool contract and not for users who are interacting with the liquidity pool through the exchange website (so through the router).

Suppose a user wants to mint liquidity in a small amplified pool with reserves X and Y through the pool contract directly. When the user initiates the transaction, it appears as a pending transaction in the mempool. A malicious actor can then place their first transaction to execute before the victim's transaction and their second transaction to execute after it (i.e., initiate a sandwich attack). In their first transaction, the attacker swaps out all X out of the pool and adds unbalanced liquidity (as described in [7]) by minting directly through the pool contract. This results in a situation where the real reserve ratio stays the same, but the virtual reserve ratio gets unbalanced. Then, the victim's transaction gets executed. Afterwards, the second attacker transaction happens, where the attacker removes all their liquidity from the pool using the burn function in the pool contract. Finally, the attacker uses the unbalanced ratio to execute a swap which results in them effectively stealing the victim's funds. As much as 12.69% of the victim's funds can be stolen in an optimal situation for the attacker [7].

4 Possible mitigations

Sandwich attacks are based on frontrunning and backrunning, i.e., transaction reordering. Therefore, any mitigation against transaction reordering would logically also mitigate sandwich attacks. First, existing mitigations in literature are presented and discussed. Their viability for the situation at hand is considered. Then follows Kyber's current mitigation against the problem in Kyber DMM. Finally, we suggest applying Kyber's mitigation in the pool contract in addition to the router.

4.1 Mitigations in literature

In literature, a number of mitigations against transaction reordering exist [29] [8]. These mitigations mainly prevent frontrunning, which renders backrunning useless since it relies on the slippage the frontrunning transaction causes for reaping the benefits of the victim's transaction. An example of a mitigation against transaction reordering is to make the transaction confidential [30]. Namely, the fact that (pending) transactions are transparent on the blockchain makes it possible for attackers to pinpoint a transaction that will be profitable. Specifically, they are interested in the amount transferred. If the attacker doesn't know the amount transferred, it becomes impossible for them to pinpoint transactions that are worth the hassle. However, this mitigation does not seem realistic for the Kyber DMM vulnerability as Kyber would need to move to a confidential blockchain or create their own. Moreover, this goes against one of the premises in the blockchain concept known as transparency.

Another mitigation is to sequence the transactions such that they cannot be tampered with by malicious actors. 0x Protocol implements this by keeping an order book off-chain [31]. This results in centralization to some degree but does help mitigate the issue as transaction reordering by malicious actors becomes impossible. However, the order book is kept by a third party whom users will need to trust when using the protocol. The order book hosts are known as Relayers, and theoretically they can reorder transactions maliciously. Finally, this mitigation goes against the decentralization premise in DeFi.

A mix of sequencing and confidentiality is also possible. A known example is the commit/reveal technique as depicted in Figure 4 [32]. With the implementation of this technique, a transaction's details only become known once the transaction is certain to execute in some order. Namely, the transaction remains hidden during the commitment phase and is made visible in the reveal phase. This ensures that transaction reordering cannot happen, as the order has already been established when it is a pending transaction. This eliminates the possibility of sandwich attacks as well, as it is based on transaction reordering. This mitigation calls for major structural changes in Kyber DMM. In addition, it reduces transparency, as transactions remain hidden for a period of time. This goes against the premise of transparency on the blockchain.

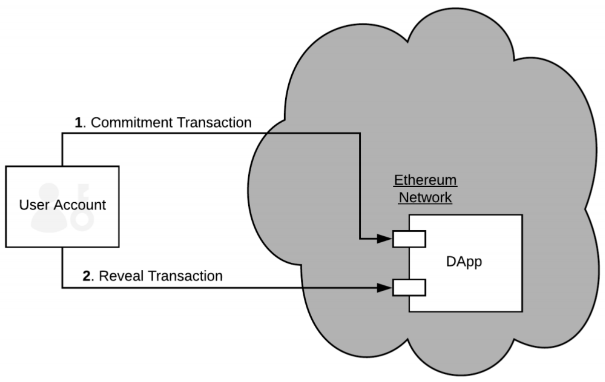

Figure 4: A schematic of the commit/reveal technique [32]

Finally, an improvement over the commit/reveal technique was brought out and it is called Submarine Send [33]. During the commit phase, Submarine Send hides the contract address in addition to the transaction until its place in the order has been established. It also ensures that the sender had enough collateral at the time of placing the transaction. This is precisely the improvement over the classic commit-reveal technique. This mitigation, similar to the original commit/reveal, reduces transparency for the same reason: transactions remain hidden for a certain duration. Therefore, this mitigation also goes against the transparency premise on the blockchain.

4.2 Kyber's current mitigation

The security audit by ChainSecurity targets only the addLiquidity() function of the router in their report [7]. Kyber implemented a degree of slippage protection by correcting their code to allow the liquidity provider to set an upper and lower bound to the virtual reserve ratio [25]. Particularly, they added two extra parameters to the function used when adding liquidity in the contract with file name "DMMRouter02.sol" (located in the periphery folder). One parameter specifies the lower bound, while the other is for the upper bound. This nullifies the effect of the attacker's first transaction, in which they swap all of one token out of the pool and add unbalanced liquidity. It's still possible to do this, but the victim's transaction will not happen because of the slippage protection. However, this mitigation has only been implemented in the router. Therefore, only users interacting with the pool contract through the DMM exchange website are protected. This means that users interacting with the pool contract directly remain unprotected. In their security audit, ChainSecurity mentions that the pool contract is still vulnerable to sandwich attacks after Kyber implemented the mitigation in the router.

4.3 Suggestion to broaden the mitigation

To protect these users as well, a similar code correction can be implemented in the pool contract. The counterpart to adding liquidity in the pool contract is the mint function. If the same slippage protection is applied here, the attacker's first transaction will be nullified in the same way as in the router. This will result in users transferring funds through the pool contract to also be protected. Only discouraging users from using this method to execute transactions is not enough, as this method can be quite crucial in times when the DMM website is unreachable.

First, a test needs to be written to determine that the current version of the DMMPool allows for sandwich attacks. The full test code is present in the following github repository [28] with file name: "sandwichAttack2.js". Executing the test code will outline details of the balances and reserves. The specific test case as it is in the repository can be summarised as follows (changing the numbers will affect the attacker's profits):

1. **Setup:**
   (a) set victim balance: 100 $\texttt{token}_0$ and 10000 $\texttt{token}_1$ ($20,000)
   (b) set attacker balance: 100 $\texttt{token}_0$ and 10000 $\texttt{token}_1$ ($20,000)
   (c) set pool reserves: 1 $\texttt{token}_0$ and 100 $\texttt{token}_1$ ($200)
   (d) set amplification factor to 5
2. **First attacker transaction**
   (a) swap 1 $\texttt{token}_0$ from pool for 125.12 $\texttt{token}_1$ to pool
   (b) add liquidity: 2.25 $\texttt{token}_0$
3. **Victim transaction**
   (a) add liquidity: 99 $\texttt{token}_0$ and 9,900 $\texttt{token}_1$. Here, the transaction should be reverted
4. **Second attacker transaction**
   (a) remove own liquidity: 2.25 $\texttt{token}_0$ and 225.12 $\texttt{token}_1$
   (b) swap 44.16 $\texttt{token}_0$ for 5,500.24 $\texttt{token}_1$
5. **Results**
   (a) attacker balance: 56.82 $\texttt{token}_0$ and 15,600.24 $\texttt{token}_1$ ($21,282.82)
   (b) pool reserves: 143.17 $\texttt{token}_0$ and 4,399.75 $\texttt{token}_1$ ($18,717.17)
   (c) victim balance: 1.01 $\texttt{token}_0$ and 100.01 $\texttt{token}_1$ ($200.01)

The attacker starts with $20,000 and ends up with $21,282.82.

$$\begin{gathered} 21,282.82/20,000 \approx 1.0641 \\ (1.0641 - 1) \times 100\% = 6.41\% \end{gathered} \tag{3}$$

Thus, in this example, the attacker makes a profit of approximately 6.41%. The final swap makes use of the unbalanced virtual reserve ratio induced in the first attacker transaction. After the victim adds their liquidity to the pool and the attacker removes theirs, the virtual reserve ratio becomes 175.9:27,490.19 (1:156.28). After the final swap, it becomes 220.07:21,989.94 (it gets restored to the original 1:100 ratio).

With the current code, the transaction does not get reverted and therefore the test case fails. For the test to pass, consider the following code correction in the mint function. The function should have two additional parameters which specify the lower and upper bound for the virtual reserves' ratio respectively. The current reserve ratio can be easily computed by the existing ReserveData struct in the DMM pool source code, which one can find in [25] with file name DMMPool.sol. The code with the correction should revert the victim transaction as specified in the list above if the ratio exceeds the bounds set by the user. However, just adding the parameters to the function causes the code to not compile at all. Therefore, we suggest Kyber to implement this idea in the mint function and we do not implement the code correction in this paper.

5 Responsible research

Ethics and reproducibility are an integral part of any research, and this research is no different. This section discusses potential negative real-world implications if the warnings are not followed as well as a reasoning for why this research is reproducible and responsible.

This research has expressed that performing a sandwich attack in Kyber DMM is possible and has shown how this attack can be mitigated. An outline of the attack has also been given. As such, a bad actor can use what is mentioned in this paper for personal financial gain, as the mitigation is not implemented yet as of writing this paper. The steps for executing the attack mentioned in this paper should only be used for testing purposes and not for stealing funds from a liquidity pool. In this paper, it is shown where the pool contract can be tested against sandwich attacks safely and without consequences, so readers who want to recreate this attack should do so on the mentioned platform or in a similar environment in a way that will not impact real users of the protocol. Once again, in short: the DMM protocol is an active system which is used for financial transactions. The attack and/or findings in this paper can be exploited by malicious actors. Perhaps the suggestion and test code should be shared with the developers of the system to improve it. This part of the research is reproducible, as the platform and methodology for arriving at the results have been clearly stated. The implementation of the DMM is open source, so any use of the code in this research is no infringement on copyright laws and falls under fair use.

Other than the proposed mitigation, this paper largely contains results from a literature study where the literature used consists of scientific papers, whitepapers, security audits and Kyber's own blog posts. Therefore, the sources used are reliable. This part of the research is easily reproducible thanks to the references in the text itself and the list of references. It also conforms to ethical standards and is thereby considered to be responsible research.

6 Discussion and Limitations

All in all, this paper has concluded that there are many possible mitigations against sandwich attacks. Some of these have already been implemented, for example: an off-chain order book for sequencing transactions in 0x Protocol [31]. The mitigations in literature however are not suitable for the problem at hand, which is sandwich attacks in Kyber DMM. A more plausible solution is to implement slippage protection in the minting function in addition to the add_liquidity function in the router, which has been suggested in this paper. Slippage protection already has been implemented in the function used when adding liquidity through the router. The conclusion is that the same slippage protection can be applied in the pool contract, mitigating the attack similarly for vulnerable users.

The test case that was explained in section 4.3 fails as of this moment, which is good. The code correction has not been implemented yet, so this confirms that the code as it is now is vulnerable to sandwich attacks if the victim mints liquidity through the pool contract directly. The code correction has not been implemented in the mint function. A capable developer could implement the correction here. The correction will lead to the community and outsiders gaining more trust in the security of Kyber's DMM and it will also lead to the protection and peace of mind for users of the protocol. The conclusion was reached by means of a literature study, considering scenarios in which a sandwich attack would be possible, discussing the matter with the Kyber team and experimentation with the source code.

7 Conclusions and Future Work

This work has answered the research question: How can one mitigate sandwich attacks in Kyber DMM? Since sandwich attacks are based on transaction reordering, a mitigation against this would mitigate sandwich attacks as well. Possible mitigations known in literature are confidentiality, transaction sequencing and the (enhanced) commit/reveal technique. Kyber's case is slightly more special, as the vulnerability arises in the virtual reserve which exists because of the amplification factor. Kyber's current mitigation only exists in the router, so the vulnerability still exists if users mint liquidity via the pool contract directly. This paper has compiled existing mitigations against the sandwich attack, showed by means of modifying an existing test case that the users interacting with the pool contract directly are indeed vulnerable to sandwich attacks, and featured a suggestion to broaden the existing mitigation to encompass the mint function in the pool contract as well. The function used when adding liquidity in the router accepts two parameters, one to specify the lower bound for the virtual reserve ratio and one to specify the upper bound. Specifically, the suggestion is to implement a similar code correction in the mint function of the pool contract. This mitigation renders the first attacker transaction useless, as the victim's transaction gets cancelled in the event of exceeding the lower or upper bounds as specified by the victim. In this scenario, the attacker becomes the victim as they pay gas fees and essentially get nothing in return.

Only users who add liquidity through the pool contract directly are vulnerable to a sandwich attack. This way of adding liquidity is discouraged by Kyber and the only situation where such an act makes sense is when the interface is down. Thus, there is a low chance for this attack to occur. This vulnerability would therefore not be considered as a severe one. However, the fact that this may come in handy for users who want to trade when the website is down remains true. Users might simply prefer to add liquidity through minting as well. Therefore, the code should also be corrected in the pool contract itself so that these users are protected when adding liquidity.

This work has only suggested to broaden an existing mitigation and has not featured a solution to the issue. The suggestion has not been implemented in this research, so a suggestion for future work would be to implement this correction in the pool contract. Another suggestion would be to explore a solution that Kyber can implement for their DMM. Furthermore, future work can be done on problems related to the DMM which are present in the latest security audit but have not been resolved yet [7]. Even though it has launched on the mainnet, the DMM protocol is a fairly new system and is still in the beta phase as of writing. Thus, another suggestion would be to further research what undiscovered problems are present in this system.

References

[1] (). "DeFi pulse — the DeFi leaderboard — stats, charts and guides," DeFi Pulse, [Online]. Available: https://defipulse.com (visited on 06/23/2021).

[2] L. Luu and Y. Velner. (Aug. 27, 2017). "Kyber network whitepaper - whitepaper.io," [Online]. Available: https://whitepaper.io/document/43/kyber-network-whitepaper (visited on 04/23/2021).

[3] "Expert security audit of Kyber Network v2," ChainSecurity, Jun. 29, 2018. [Online]. Available: https://github.com/KyberNetwork/smart-contracts/blob/e4d3ae21e063bfd65c4621197687334527eb54dc/audits/kyberV2Audit/ChainSecurity_KyberNetwork_Public.pdf (visited on 04/25/2021).

[4] "Kyber.Network smart contracts audit report," BlockchainLabs, 2018. [Online]. Available: https://github.com/KyberNetwork/smart-contracts/blob/e4d3ae21e063bfd65c4621197687334527eb54dc/audits/kyberV1Audit2/KyberNetwork%20BlockchainLabs%20Audit%20Report.pdf (visited on 04/25/2021).

[5] "Kyber Network Smart Contracts Security analysis," SmartDec, Dec. 2, 2019. [Online]. Available: https://github.com/KyberNetwork/smart-contracts/blob/e4d3ae21e063bfd65c4621197687334527eb54dc/audits/APRAudit/SmartDecAudit.pdf (visited on 04/25/2021).

[6] "Security audit of KyberNetwork's Smart Contracts," ChainSecurity, Jan. 9, 2019. [Online]. Available: https://github.com/KyberNetwork/smart-contracts/blob/e4d3ae21e063bfd65c4621197687334527eb54dc/audits/kyberV3Audit/ChainSecurity_Kyberv3.pdf (visited on 04/25/2021).

[7] "Code Assessment of the Dynamic Market Maker Smart Contracts," ChainSecurity, Apr. 23, 2021. [Online]. Available: https://chainsecurity.com/wp-content/uploads/2021/04/ChainSecurity_KyberNetwork_DMM_Dynamic-Market-Making_Final.pdf (visited on 05/04/2021).

[8] L. Zhou, K. Qin, C. F. Torres, D. V. Le, and A. Gervais, "High-Frequency Trading on Decentralized On-Chain Exchanges," *arXiv:2009.14021 [cs]*, Sep. 29, 2020. [Online]. Available: http://arxiv.org/abs/2009.14021 (visited on 05/16/2021).

[9] K. Qin, L. Zhou, and A. Gervais, "Quantifying Blockchain Extractable Value: How dark is the forest?" *arXiv:2101.05511 [cs]*, Jan. 21, 2021. [Online]. Available: http://arxiv.org/abs/2101.05511 (visited on 05/09/2021).

[10] S. M. Werner, D. Perez, L. Gudgeon, A. Klages-Mundt, D. Harz, and W. J. Knottenbelt, "SoK: Decentralized Finance (DeFi)," *arXiv:2101.08778 [cs, econ, q-fin]*, Apr. 30, 2021. [Online]. Available: http://arxiv.org/abs/2101.08778 (visited on 05/09/2021).

[11] J. Xu, N. Vavryk, K. Paruch, and S. Cousaert, "SoK: Decentralized Exchanges (DEX) with Automated Market Maker (AMM) protocols," *arXiv:2103.12732 [cs, q-fin]*, Apr. 19, 2021. [Online]. Available: http://arxiv.org/abs/2103.12732 (visited on 05/06/2021).

[12] D. A. Zetzsche, D. W. Arner, and R. P. Buckley, "Decentralized Finance," *Journal of Financial Regulation*, vol. 6, no. 2, pp. 172–203, Sep. 20, 2020, ISSN: 2053-4841. DOI: 10.1093/jfr/fjaa010. [Online]. Available: https://doi.org/10.1093/jfr/fjaa010 (visited on 04/25/2021).

[13] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system — bitcoin paper," Aug. 21, 2008. [Online]. Available: https://www.bitcoinpaper.info/bitcoinpaper-html/ (visited on 06/02/2021).

[14] (). "Script - Bitcoin Wiki," [Online]. Available: https://en.bitcoin.it/wiki/Script (visited on 06/02/2021).

[15] (Feb. 9, 2021). "Ethereum whitepaper," ethereum.org, [Online]. Available: https://ethereum.org (visited on 05/29/2021).

[16] (). "Solidity — Solidity 0.8.4 documentation," [Online]. Available: https://docs.soliditylang.org/en/v0.8.4/ (visited on 06/02/2021).

[17] S. Wang, Y. Yuan, X. Wang, J. Li, R. Qin, and F.-Y. Wang, "An Overview of Smart Contract: Architecture, Applications, and Future Trends," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, ISSN: 1931-0587, Jun. 2018, pp. 108–113. DOI: 10.1109/IVS.2018.8500488. (visited on 04/25/2021).

[18] A. Goodpasture. (May 11, 2021). “What is the ethereum gas fee and how much does it cost?” Market Realist, [Online]. Available: https://marketrealist.com/p/ethereum-gas-fee-explained/ (visited on 06/25/2021).

[19] D. Perez, S. M. Werner, J. Xu, and B. Livshits, “Liquidations: DeFi on a Knife-edge,” *arXiv:2009.13235 [q-fin]*, Apr. 5, 2021. [Online]. Available: http://arxiv.org/abs/2009.13235 (visited on 04/19/2021).

[20] Kyber. (Jun. 8, 2021). “Kyber 3.0: Architecture revamp, dynamic MM, and KNC migration proposal,” Medium, [Online]. Available: https://blog.kyber.network/kyber-3-0-architecture-revamp-dynamic-mm-and-knc-migration-proposal-acae41046513 (visited on 04/29/2021).

[21] F. Schär. (). “Decentralized finance: On blockchain- and smart contract-based financial markets,” [Online]. Available: https://research.stlouisfed.org/publications/review/2021/02/05/decentralized-finance-on-blockchain-and-smart-contract-based-financial-markets (visited on 04/21/2021).

[22] (). “Top DEX exchanges by trading volume - CoinGecko,” CoinGecko, [Online]. Available: https://www.coingecko.com/en/dex (visited on 06/23/2021).

[23] Kyber. (Jun. 8, 2021). “Kyber DMM beta is live!” Medium, [Online]. Available: https://blog.kyber.network/kyber-dmm-beta-is-live-b6bdd18d0dde (visited on 04/29/2021).

[24] H. Adams. (). “Uniswap whitepaper,” HackMD, [Online]. Available: https://hackmd.io/@HaydenAdams/HJ9jLsfTz (visited on 05/21/2021).

[25] Kyber, *dynamic-amm/smart-contracts*, Apr. 28, 2021. [Online]. Available: https://github.com/dynamic-amm/smart-contracts (visited on 05/26/2021).

[26] A. Nguyen, L. Luu, and M. Ng, *Dynamic Automated Market Making*, Feb. 2021. [Online]. Available: https://files.kyber.network/DMM-Feb21.pdf (visited on 05/06/2021).

[27] Kyber. (). “DMM Exchange,” [Online]. Available: https://dmm.exchange/#/about (visited on 06/23/2021).

[28] A. A. Yüksel, *Kyber DMM smart contracts*, Jun. 15, 2021. [Online]. Available: https://github.com/Kiwi-42/smart-contracts (visited on 06/15/2021).

[29] P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels, “Flash Boys 2.0: Frontrunning, Transaction Reordering, and Consensus Instability in Decentralized Exchanges,” *arXiv:1904.05234 [cs]*, Apr. 10, 2019. arXiv: 1904.05234. [Online]. Available: http://arxiv.org/abs/1904.05234 (visited on 04/28/2021).

[30] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, “Bulletproofs: Short Proofs for Confidential Transactions and More,” in *2018 IEEE Symposium on Security and Privacy (SP)*, ISSN: 2375-1207, May 2018, pp. 315–334. DOI: 10.1109/SP.2018.00020. (visited on 06/05/2021).

[31] W. Warren and A. Bandeali. (Feb. 21, 2017). “0x: An open protocol for decentralized exchange on the ethereum blockchain,” The Whitepaper Database, [Online]. Available: https://www.allcryptowhitepapers.com/0x-whitepaper/ (visited on 06/05/2021).

[32] S. Eskandari, S. Moosavi, and J. Clark, “SoK: Transparent Dishonesty: front-running attacks on Blockchain,” *arXiv:1902.05164 [cs]*, Apr. 9, 2019. [Online]. Available: http://arxiv.org/abs/1902.05164 (visited on 05/06/2021).

[33] L. Breidenback, P. Daian, A. Juels, and F. Tramèr. (Aug. 28, 2017). “To Sink Frontrunners, Send in the Submarines,” Hacking Distributed, [Online]. Available: https://hackingdistributed.com/2017/08/28/submarine-sends/ (visited on 06/05/2021).