# SoK: Mitigation of Front-running in Decentralized Finance

Carsten Baum[1], James Hsin-yu Chiang[2], Bernardo David[3], Tore Kasper Frederiksen[4], Lorenzo Gentile[3],

[1] Aarhus University, Denmark
cbaum@cs.au.dk
[2] Technical University of Denmark, Denmark
jchi@dtu.dk
[3] IT University of Copenhagen, Denmark
bernardo@bmdavid.com, lorg@itu.dk
[4] Alexandria Institute, Denmark
tore.frederiksen@alexandra.dk

**Abstract.** Front-running is the malicious, and often illegal, act of both manipulating the order of pending trades and injecting additional trades to make a profit at the cost of other users. In decentralized finance (DeFi), front-running strategies exploit both public knowledge of user trades from transactions pending on the network and the miner's ability to determine the final transaction order. Given the financial loss and increased transaction load resulting from adversarial front-running in decentralized finance, novel cryptographic protocols have been proposed to mitigate such attacks in the permission-less blockchain setting. We systematize and discuss the state-of-the-art of front-running mitigation in decentralized finance, and illustrate remaining attacks and open challenges.

## 1 Introduction

Specific instances of front-running in decentralized finance (DeFi) were first quantified by Daian et al. [17] and systematized by Eskandari et al. [23]. Besides imposing a financial penalty on honest users, front-running can also degrade the performance of blockchain networks, as recently observed on the Avalanche blockchain [3]. In order to evaluate the efficacy of front-running mitigation techniques, we first formulate the set of adversarial powers which permit front-running strategies to be exploited: concretely, if users submit their intended interaction to a pool of pending transactions, the front-running adversary has the ability to:

1. Append pending transactions to the blockchain.
2. Infer user intentions from pending transactions and blockchain state.

In this work, we describe common **front-running attacks** (§2) and assess three front-running **mitigation categories** (§3) for their isolated and combined efficacy in neutralizing front-running (Figure 1). We introduce a speculative sandwich attack on input batching techniques (§3.2), which can be mitigated with private user balances and secret input stores (§3.3).

| **Adversarial power** | §3 **Mitigation** | |
|---|---|---|
| 1. Transaction sequencing | §3.1 Fair ordering | |
| | §3.2 Batching of blinded inputs | Commit & reveal |
| 2. Inference of user intent | | Input aggregation |
| | §3.3 Private user balances & secret input store | |

Fig. 1: Overview of mitigation techniques

**Fair ordering** (§3.1), implemented at the consensus protocol layer, ensures that the local receipt-order of gossiped transactions seen by a node is consistent with the final transaction ordering in the blockchain. We observe that *fair ordering* effectively mitigates the miner's ability to freely sequence transactions, but introduces a front-running adversary which rushes the network.

**Batching of blinded inputs** (§3.2) replaces the *sequential* model of DeFi interaction with a *round-based* one, where user inputs are blinded in each round to ensure input independence, thereby thwarting front-running strategies that rely on prior knowledge of other users' intentions. However, if user balances are public, the input may still be partially inferred when the valid user's input space is constrained by it's balance: here, we contribute a novel, *speculative* front-running attack that exploits the *direction* of an automated market maker (AMM) swap, leaked from the victim's public balance. Furthermore, we highlight differences between *commit & reveal* and *input aggregation* approaches to batching of blinded inputs (Figure 2). In commit & reveal schemes, user inputs are revealed *individually*: Although front-running in the specific round is no longer possible, they necessarily leak information about the subsequent balance-update for each participating user, even if the user balances are private. If the taint of private balances is sufficiently strong, this can allow the front-running adversary to infer the users future inputs (e.g. the intended AMM swap direction).

| | | User balance & input store | |
|---|---|---|---|
| | | Public | **Private, secret** |
| Batching of blinded inputs | Commit & reveal | Speculative Sandwich Attacks | Taint of user balances |
| | **Input aggregation** | | - |

Fig. 2: Efficacy: batching of blinded inputs.

**Private user balances** (§3.3) are thus necessary to prevent the leakage of the valid user input space from balances and application state. Although DeFi state must generally remain public to retain its utility [2], we show that it is necessary to shield certain fragments thereof which explicitly reveal future user intent. **Secret input stores** (§3.3) protect inputs that are evaluated by the application after a time delay [44] or, in the case of order books, whenever a match with other user inputs [24,7] can be found.

## 2 Front-running attacks

**AMM sandwich:** We briefly summarize the functionality of constant product AMM's, namely, a liquidity pool holding token balances, $r_0$ and $r_1$, of two different token types, $\tau_0$ and $\tau_1$ respectively, s.t. $r_0 \cdot r_1$ is *always* constant when swaps are being carried out between $\tau_0$ and $\tau_1$. A user swaps units of $\tau_0$ for units of $\tau_1$ by authorizing a *left swap* action $\mathsf{SL}(v : \tau_0, w : \tau_1)$. Here, the user is sending $v : \tau_0$ to the AMM in return for at least $w : \tau_1$ (swap limit). For this left swap to be valid, the product of the reserves must be maintained. Thus, the following relation between initial and updated reserves must hold: $r_0 \cdot r_1 = (r_0 + v) \cdot (r_1 - w')$, where $w' \geq w$ and $w'$ represents the units of $\tau_1$ that the user actually gets. We refer $w$ as the swap *limit*. A *right swap* of $\mathsf{SR}(v : \tau_0, w : \tau_1)$ follows similarly: the user sends $w : \tau_1$ for at least $v : \tau_0$ in return such that $r_0 \cdot r_1 = (r_0 - v') \cdot (r_1 + w)$ and $v' \geq v$ where $v'$ represents the units of $\tau_0$ received. Constant product AMM's exhibit *slippage*: subsequent swaps in the same direction exhibit decreasing exchange rates.

User swaps can be "sandwiched", exploiting slippage for the gain of the attacker. Consider a left swap $\mathsf{A} : \mathsf{SL}(\mathsf{v_A} : \tau_0, \mathsf{w_A} : \tau_1)$ submitted by user $\mathsf{A}$. A front-run swap by attacker $\mathsf{M}$ in the same direction reduces the exchange rate for the subsequent victim swap: a final back-run swap by $\mathsf{M}$ in the opposing direction then profits from an improved exchange rate.

$$\mathsf{M} : \mathsf{SL}(\mathsf{v^f_M} : \tau_0, \mathsf{w^f_M} : \tau_1) \quad \mathsf{A} : \mathsf{SL}(\mathsf{v_A} : \tau_0, \mathsf{w_A} : \tau_1) \quad \mathsf{M} : \mathsf{SR}(\mathsf{v^b_M} : \tau_0, \mathsf{w^b_M} : \tau_1)$$

Optimal front-run $(\mathsf{v^f_M}, \mathsf{w^f_M})$ and back-run $(\mathsf{v^b_M}, \mathsf{w^b_M})$ parameters are a function of the victim's swap, inferred from the pending victim transaction gossiped across the network [5].

We illustrate a step-wise execution of a sandwich in Figure 3 and introduce notation for user and AMM state proposed in [4] for this purpose. The wallet of $\mathsf{A}$ is modelled as the term $\mathsf{A}[v_i : \tau_0, ..., v_n : \tau_n]$, where $v_0, ..., v_n$ are the respective balances of token types $\tau_0, ..., \tau_n$. The state of an AMM holding token types $\tau_0$ and $\tau_1$ is given by its reserve balances $(r_0 : \tau_0, r_1 : \tau_1)$. Thus, we express the system state as a composition of wallets and reserve balances.

$$\mathsf{A}[v : \tau] \mid (r_0 : \tau_0, r_1 : \tau_1)$$

Let the initial AMM balance be $(100 : \tau_0, 100 : \tau_1)$. User $\mathsf{A}$ wishes to perform the swap $\mathsf{A} : \mathsf{SL}(15 : \tau_0, 10 : \tau_1)$. For simplicity, we assume unit values of $\tau_0$ and $\tau_1$ to be equal: given the ratio of AMM reserves is 1, there is no arbitrage opportunity to be exploited [4]. If $\mathsf{A}$'s order is executed immediately, $\mathsf{A}$ receives $13 : \tau_1$ for the $15 : \tau_0$ it sends to the AMM. Instead, however, if the user swap is sandwiched by attacker $\mathsf{M}$ (Figure 3), $\mathsf{A}$ only obtains the minimum amount $10 : \tau_1$, implying a reduction of $3 : \tau_1$. Note that the reserve product is maintained at each execution step and that the sandwich execution preserves the initial reserve ratio: the attack leaves no arbitrage opportunity unexploited. The attacker $\mathsf{M}$'s profit of 5 units of $\tau_0$ (or $\tau_1$) is optimal [5]: $\mathsf{A}$ receives the minimum amount possible, namely its swap limit.

**Scheduled AMM sandwich:** For certain AMM variants, the knowledge of the user's intent to perform a swap can be directly inferred from the blockchain state. Paradigm [44] propose scheduled AMM swaps, or more generally, *scheduled*

$$\begin{aligned}
&\mathsf{A}[15:\tau_0] \mid \mathsf{M}[15:\tau_0, 10:\tau_1] \mid (100:\tau_0, 100:\tau_1)\\
\xrightarrow{\mathsf{M}:\mathsf{SL}(15:\tau_0,13:\tau_1)}\ &\mathsf{A}[15:\tau_0] \mid \mathsf{M}[23:\tau_1] \mid (115:\tau_0, 87:\tau_1)\\
\xrightarrow{\mathsf{A}:\mathsf{SL}(15:\tau_0,10:\tau_1)}\ &\mathsf{A}[10:\tau_1] \mid \mathsf{M}[23:\tau_1] \mid (130:\tau_0, 77:\tau_1)\\
\xrightarrow{\mathsf{M}:\mathsf{SR}(30:\tau_0,23:\tau_1)}\ &\mathsf{A}[10:\tau_1] \mid \mathsf{M}[30:\tau_0] \mid (100:\tau_0, 100:\tau_1)
\end{aligned}$$

Fig. 3: Sandwich attack

*inputs*. Let $\mathsf{A} : \mathsf{SL}(15 : \tau_0, 10 : \tau_1, \mathsf{r})$ be a swap that is not executed immediately, but scheduled for evaluation together with the first user-AMM interaction *following* blockchain round $\mathsf{r}$, thus requiring no further interaction from $\mathsf{A}$. Since *scheduled* orders are stored in the AMM smart contract and evaluated at the beginning of a known round, the sandwich attack strategy can be exploited, albeit over two block rounds [44]: the front-run is sequenced at the end of round $\mathsf{r}$ and the back-run as the first newly submitted swap of round $\mathsf{r} + 1$.

**Generalized front-run attacks:** In decentralized finance, actions exist which are *profitable* for the authorizing user, but which can also be performed by any other agent with a sufficient balance. In the permissionless blockchain setting, *generalized front-runners*, a term coined by Daian [37], are automated agents that identify profitable, pending transactions, which can be authorized by *any* user, and simply replicate these with their own account, thereby depriving the original transaction submitter of it's profit. Since the security of DeFi applications rely on rational agents to solve for profitable arbitrage [46,43,20] and liquidation [40] strategies, the presence of generalized front-running threatens to restrict such opportunities to agents colluding with miners.

## 3 Mitigation categories

### 3.1 Fair ordering

A recent line of research [33,30,31] has formalized an intuitive notion of $\gamma$-*receipt-order-fairness*: given two distinct transactions $\mathsf{tx}$ and $\mathsf{tx}'$ broadcast by users, receipt-order-fairness of a consensus protocol ensures that $\mathsf{tx}$ will be finalized prior to $\mathsf{tx}'$ if a $\gamma$ fraction of network nodes receives $\mathsf{tx}$ prior to $\mathsf{tx}'$. However, Kelkar et al. [30] show that even if all nodes agree on the relative order in which any *pair* of transactions were first observed at the gossip stage, a global transaction ordering of all transactions consistent with the local view of pair-wise orderings is not always possible (Condorcet Paradox). Instead, a weaker notion of $\gamma$-*batch-order-fairness* is realized in [31], where $\mathsf{tx}$ will be sequenced prior to or in the same block as $\mathsf{tx}'$ if a $\gamma$ node fraction receives $\mathsf{tx}$ first.

**Front-running despite fair ordering:** Although order fairness removes the miner or block-round leader's privilege to sequence transactions, it introduces a new front-running adversary: here, a *rushing* agent that observes a pending victim transaction $\mathsf{tx}$ *before* it has been received by a $\gamma$ fraction of nodes can send its front-running $\mathsf{tx}'$ to a $\gamma$ threshold of nodes *just before* $\mathsf{tx}$ is received by the same network fraction. Whereas in the standard setting the miner or round leader incurs no additional cost for front-running victims, a non-trivial communication cost is now imposed on the rushing adversary. Still, since order-fairness does not *eliminate* front-running attacks, the motivation for stronger front-running mitigation properties remains.

## 3.2 Batching of blinded inputs

Batching of blinded inputs is a technique to ensure 1) the independence between user inputs and 2) the prevention of any adversarial sequencing of inputs. Interactions occur in rounds: in each, inputs are committed during the *input-phase*, followed by an *output phase* where the application state is updated after evaluating user inputs with valid parameters. The *collection* of inputs can occur in a smart contract or by a committee executing a cryptographic protocol which authorizes the distribution of funds from a smart contract in the output phase. The update of the application state following each round can result from the evaluation of valid inputs in *randomized* order or an application-specific *aggregation* thereof: for example, a subset of submitted AMM swaps can be aggregated into a single resulting swap.

| | | Input independence | Input privacy | Open challenges |
|---|---|---|---|---|
| Commit & reveal | Hash commitments* | - | - | *Output bias* |
| | Timed commitments* | ● | - | *Delay parameters* |
| | Threshold encryption** | ● | - | *Honest majority* |
| | Secure multi-party computation** | ● | - | *Honest majority* |
| Input aggregation | | ● | ● | *Abort penalty* |
| | Homomorphic encryption** | ● | ● | *Efficiency* |

Fig. 4: Batching of blinded inputs sent to a smart contract* or committee**

In batching of blinded inputs, we distinguish between **commit & reveal** and **input aggregation** (fig. 4). Both schemes commit inputs in the input-phase of each round, thereby ensuring input independence. However, while input aggregation keeps the users' input private indefinitely, commit & reveal schemes leak individual user inputs when commitments are opened, thereby offering no *input privacy* by definition. Input privacy is necessary to prevent front-running in *subsequent* interaction rounds: past inputs leak information about updates to private balances (§3.3), which in turn can be exploited by front-runners, as balances constrain the valid user input space.

$$\text{Past user inputs} \xrightarrow{\text{reveal}} \text{Private user balances} \xrightarrow{\text{reveal}} \text{Future user inputs}$$

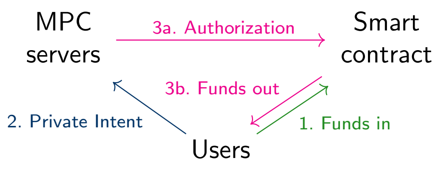
In contrast, input aggregation only outputs the application state update: for aggregated AMM swaps, only reserve updates are revealed, and updates to user balances remain private, if private balances are supported. Naturally, input aggregation can only offer input privacy up to the input batch size.

**Commit & reveal:** Although *hash commitments* collected by a smart contract may appear to be an obvious approach to implement the commit & reveal functionality, they suffer from *output bias*, as the adversary can selectively refrain from opening its commitment.

*Time-lock puzzles* [41] or *timed commitments* [11] generated by users and sent to a smart contract promise to eliminate output bias, since the adversary's commitment can be force-opened after a delay, guaranteeing the inclusion of its input in the output-phase. However, in the worst case, each user time-locked input must be solved separately by a constant number of squaring operations in a randomly sampled group, rendering the approach impractical for larger batches of time-locked inputs. To this end, Doweck and Eyal propose multi-party timed commitments [22] constructed from El-gamal encryption with a randomly sampled public key of a small group size. All inputs are encrypted with a random public key and opened when the private key is discovered by brute-force. However, key space exploration is parallelizable, such that no lower bound of operations can be guaranteed. Furthermore, it remains an open challenge to match time-lock puzzle parameters to real-world delays which depend on assumed gate speeds used in practice.

*Threshold encryption* [21] can realize a commit & reveal scheme with the assumption of an *honest majority* committee holding trapdoor information of the encrypted inputs. In a setup phase, a public key is produced by the committee, with which users encrypt their inputs in each round. Encrypted inputs are posted to a smart contract which verifies the decryption by the committee during each reveal-phase.

*Secure multi-party computation* [45,27] (MPC) has been proposed [35,1] to realize a commit & reveal functionality with guaranteed input reveal in an anonymous fashion, also formalized as *anonymous committed broadcast* (ACB) in [1]. The anonymization of inputs is achieved by random *shuffling* of user inputs in an efficient manner. Here, *honest majority* MPC protocols [8,18] are favoured, as the output is guaranteed as long as the honest majority assumption holds true. To implement a DeFi application with MPC, an MPC-controlled smart contract is required, to which users send their funds prior to each round.



In the output phase of each MPC round, funds in the smart contract are redistributed to users according to the output(s) of the MPC execution. In practice, users can safely delegate the MPC execution to a group of servers [1].

**Input aggregation:** Naturally, MPC can realize any aggregation function over private user inputs, and in some instances in an efficient manner. Given the emphasis on the privacy of inputs, *dishonest majority* MPC protocols [14,10,19] are favoured, which ensure that private inputs can never be obtained by the adversary as long as a single participant remains honest. Informal proposals to implement AMM instances in a dishonest majority MPC have been proposed by Li et al. [34]. Although dishonest majority MPC can be aborted by a single dishonest party, a recent line of research [32,6,7] has realized an efficient set of protocols that identify and financially punish the aborting adversary. This achieves a weaker notion of fairness as the rational adversary is incentivized to never abort. Still, the penalty must exceed the financial *option* value of aborting in order to be effective: given that inputs are private, it remains an open research question on how to size financial penalties for identifiable abort in MPC.

Penumbra [39] proposes the use of *homomorpic encryption* to realize the secure aggregation of homomorphically encrypted AMM swap orders. The aggregated swap is then decrypted to reveal the updated AMM reserves. User balances are implemented with private coins (see §3.3), thus the privacy of the inputs are only dependent on the batch size. We note the non-trivial complexity of aggregating a batch of encrypted AMM swaps with swap limit constraints: *efficient* secure multi-party computation with fully homomorphic encryption schemes remains an open research problem [26]. In [39], consensus validators are proposed to perform the secure computation, consolidating MPC and consensus layers.

**Speculative sandwich w/public user balances:** We illustrate that batching of blinded inputs alone is not sufficient to prevent front-running attacks. Instead, speculative AMM sandwich attacks are possible in blinded input batching schemes as long as the direction of the victim swap is known by the adversary. This can be inferred from *public* user balances, as detailed in the subsequent example. Such speculative sandwich attacks on batched inputs also assume that the adversary in the permissionless setting can "isolate" a single victim's input in a given round, such that only front-run and victim transactions remain: we argue that each batching round has participant limits due to gas constraints or number of clients that MPC servers can support. Thus, the adversary can occupy any arbitrary number of user slots per round and provide invalid inputs[5] on slots not dedicated to the front-running swap. In this speculative attack, we

| Round r | | Round r+1 |
|---|---|---|
| $\mathsf{M} : \mathsf{SL}(\mathsf{v}^{\mathsf{f}}_{\mathsf{M}} : \tau_0, \mathsf{w}^{\mathsf{f}}_{\mathsf{M}} : \tau_1)$ | $\mathsf{A} : \mathsf{SL}(\mathsf{v}_{\mathsf{A}} : \tau_0, \mathsf{w}_{\mathsf{A}} : \tau_1)$ | $\mathsf{M} : \mathsf{SR}(\mathsf{v}^{\mathsf{b}}_{\mathsf{M}} : \tau_0, \mathsf{w}^{\mathsf{b}}_{\mathsf{M}} : \tau_1)$ |

Fig. 5: Speculative sandwich

assume that private AMM swaps in each blinded input batch are evaluated in a *random* order, as proposed in [34,1]. The front-running M can only speculate on achieving the correct order to execute the sandwich. Since balances are public,

[5] e.g. AMM swap parameters which cannot be executed in the current AMM state.

$\mathsf{M}$ can observe that $\mathsf{A}$'s balance of $\tau_1$ is zero: thus, $\mathsf{A}$'s submitted swap to the AMM $(\tau_0, \tau_1)$ must be in the *left* direction. $\mathsf{M}$ submits the *front-run* swap in the same direction as the victim in the initial round $\mathsf{r}$.

In the optimistic case shown in Figure 5, $\mathsf{M}$'s front-run swap is evaluated *prior* to the victim swap (in round $\mathsf{r}$), thus enabling $\mathsf{M}$ to position the profitable back-run swap in round $\mathsf{r}+1$, where all other users are prevented from submitting inputs. $\mathsf{M}$'s front-run parameters can be chosen such that the front-run swap simply does not execute should the front-run *not* be ordered prior to the victim swap in round $\mathsf{r}$, thereby aborting the attack. We refer to Appendix A for the proof that this speculative sandwich is rational for the attacker.

An execution of a speculative sandwich is shown in Figures 6 and 7: here, adversary $\mathsf{M}$ observes victim $\mathsf{A}$'s interaction with an AMM which batches blinded inputs. $\mathsf{A}$ has a public balance of $20 : \tau_0$ only, allowing $\mathsf{M}$ to infer that $\mathsf{A}$ can only perform a *left* swap from $\tau_0$ to $\tau_1$ with an input amount of at most $20 : \tau_0$. The attack strategy is executed over two subsequent rounds beginning in the initial state shown in Figure 6, where we assume unit values of $\tau_0$ and $\tau_1$ are equal.

$$\mathsf{A}[20:\tau_0] \mid \mathsf{M}[7:\tau_0, 15:\tau_1] \mid (100:\tau_0, 100:\tau_1)$$

Round $\mathsf{r}$

$$\xrightarrow{\mathsf{M}:\mathsf{SL}(7:\tau_0, 6.5:\tau_1)} \mathsf{A}[20:\tau_0] \mid \mathsf{M}[21.5:\tau_1] \mid (107:\tau_0, 93.5:\tau_1)$$

$$\xrightarrow{\mathsf{A}:\mathsf{SL}(15:\tau_0, 10:\tau_1)} \mathsf{A}[5:\tau_0, 11.5:\tau_1] \mid \mathsf{M}[21.5:\tau_1] \mid (122:\tau_0, 82:\tau_1)$$

Round $\mathsf{r}+1$

$$\xrightarrow{\mathsf{M}:\mathsf{SR}(22:\tau_0, 18:\tau_1)} \mathsf{A}[5:\tau_0, 11.5:\tau_1] \mid \mathsf{M}[22:\tau_0, 3.5:\tau_1] \mid (100:\tau_0, 100:\tau_1)$$

Fig. 6: Successful speculative sandwich

In the first round $\mathsf{r}$, $\mathsf{M}$ submits the *front-run* swap in the same direction as the victim's, with *arbitrarily chosen* input amount $7 : \tau_0$. The minimum output amount or swap limit of the front-run is then is chosen to be $6.5 : \tau_1$ such that $(100 + 7) \cdot (100 - 6.5) = 100^2$ holds: thus, if the front-run were executed in the initial state, $\mathsf{M}$ would receive *exactly* its swap limit. Since all other user orders (other than the victim swap of $\mathsf{A}$) are suppressed, there is a probability of 0.5 that the front-run is randomly evaluated *before* the victim's swap, as shown in Figure 6. The *back-run* swap of $\mathsf{M}$ in the opposing direction then follows in the subsequent round with probability 1, since $\mathsf{M}$ suppresses all user actions other than its own back-run. Assuming equal unit value of both token types, the attack profit for $\mathsf{M}$ is 3.5.

Should the front-run ordering fail (Figure 7), then $\mathsf{M}$'s front-run parameters are chosen such that the front-run swap will not execute, resulting in an abort of the speculative sandwich attack. This is due to the chosen front-run parameters: following the execution step of $\mathsf{A}$'s swap in Figure 7, the constant product invariant can only hold if $\mathsf{M}$ receives $5 : \tau_1$ for the $7 : \tau_0$ it sends: $(115 + 7) \times (87 - 5) = 100^2$. However, this contradicts $\mathsf{M}$ swap limit of $6.5 : \tau_1$, such that the front-run cannot execute in the state following $\mathsf{A}$'s swap. $\mathsf{M}$ can still perform a back-run in round $\mathsf{r}+1$, thereby restoring the initial reserve ratio and extracting an arbitrage profit of 2, which is less than in the successful speculative sandwich execution in Figure 6. Still, the speculative sandwich attack is always profitable, as shown in Appendix A.

$$\mathsf{A}[20:\tau_0] \mid \mathsf{M}[7:\tau_0, 15:\tau_1] \mid (100:\tau_0, 100:\tau_1)$$

Round $\mathsf{r}$

$$\xrightarrow{\mathsf{A}:\mathsf{SL}(15:\tau_0,10:\tau_1)} \mathsf{A}[5:\tau_0, 13:\tau_1] \mid \mathsf{M}[7:\tau_0, 15:\tau_1] \mid (115:\tau_0, 87:\tau_1)$$

$$\xrightarrow{\mathsf{M}:\mathsf{SL}(\cancel{7:\tau_0,6.5:\tau_1})} \mathsf{A}[5:\tau_0, 13:\tau_1] \mid \mathsf{M}[7:\tau_0, 15:\tau_1] \mid (115:\tau_0, 87:\tau_1)$$

Round $\mathsf{r}+1$

$$\xrightarrow{\mathsf{M}:\mathsf{SR}(15:\tau_0,13:\tau_1)} \mathsf{A}[5:\tau_0, 13:\tau_1] \mid \mathsf{M}[22:\tau_0, 2:\tau_1] \mid (100:\tau_0, 100:\tau_1)$$

Fig. 7: Aborted speculative sandwich

Importantly, if victim A's swap direction were unknown, M would have to guess the direction of the front-running swap. An incorrect guess can result in a loss for M as shown in Appendix B. Thus, we argue that private user balances are necessary for batching of blinded inputs to be effective. Furthermore, for *scheduled* AMM orders introduced in [44], private user balances remain insufficient if scheduled orders are stored in public smart contracts: we sketch a speculative sandwich attack on publicly scheduled swaps in Appendix C. Finally, we note that hash-based commit & reveal schemes permit speculative sandwich attacks even when user balances are private, as the adversary can selectively reveal the appropriate sandwich strategy which matches on the swap first revealed by the victim (Appendix D).

### 3.3 Private & secret state

As argued in §3.2, both the *aggregation* of blinded inputs and use of *private balances and secret input stores* is necessary to mitigate front-running in the current and future rounds. Whilst it may be possible to maintain the *entire* DeFi application state secretly in an MPC instance in order to prevent front-running, this will naturally reduce its utility to users in the permissionless setting. Notably, Angeris et al. [2,15] argue that both *marginal price* and *validity* of a given AMM swap order must be queryable for an AMM interaction to be meaningful. Therefore, we restrict our study of secret state in DeFi applications to *user input stores* [44,24], which maintain submitted inputs until they are evaluated or executed at a later point in time.

**Private user balances:** Private block-chain currencies and tokens have been realized with zero-knowledge proof systems: *confidential transactions* [36] shield output amounts with efficient zero-knowledge *range proofs* [13], thereby ensuring that newly created output values do not exceed those spent by the same transaction. Confidential transactions only shield output amounts: a transaction graph connecting outputs can still be inferred from public transactions on the block-chain, permitting coin taint to propagate downstream.

Z-cash [42] style *decentralized anonymous payment* (DAP) schemes break such public links between outputs, as well-formed relations between new and spent outputs are not revealed but publicly verifiable with SNARK [28,25,38,9,29] zero-knowledge proofs. DAP schemes have also been proposed for DeFi functionality in Manta [16], but here front-running is not mitigated, since the AMM reserve state is public and swap inputs are not batched. Even though swap parameters are blinded in Manta, each individual swap execution results in a *public* update of AMM reserves. Thus, the *affect* of each swap on the current AMM reserves is known, leaking exchanged amounts and permitting sandwich attack strategies.

Importantly, when implementing input batching (Figure 4) with secure computation *and* block-chains supporting private user balances, zero-knowledge proofs must be generated inside the MPC instance in order to update private user balances. Doing so *efficiently* in MPC or even fully homomorphic encryption remains on open research question.

Finally, Submarine commitments [12] propose that users can rely on k-anonymity alone to privately commit funds during the input-phase without the use of private balances. Here, users commit value to an *k-anonymized* address which can only be withdrawn by a specific smart contract after the address is revealed together with the input by the user.

**Secret input stores:** We note that shielded scheduled AMM swaps [44] or long-running order lists [24] cannot be maintained by encryption alone: encryption of a scheduled swap by a user implies its decryption at a later stage, requiring repeated user interaction, and thus defeating the purpose of scheduled inputs. Alternatively, a decryption by an honest majority committee implies that the round or block-height of the input schedule is known. Instead, we suggest a long-running MPC instance to realize secret input stores in decentralized finance. Here, stored inputs are secret shared across MPC servers: in each round, both newly submitted inputs and secretly stored inputs are secretly evaluated together to update the application state, neither being visible to the front-running adversary.

## References

1. Abraham, I., Pinkas, B., Yanai, A.: Blinder–Scalable, Robust Anonymous Committed Broadcast. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. pp. 1233–1252 (2020). https://doi.org/10.1145/3372297.3417261
2. Angeris, G., Evans, A., Chitra, T.: A Note on Privacy in Constant Function Market Makers. arXiv preprint arXiv:2103.01193 (2021), https://arxiv.org/abs/2103.01193
3. Avalanche: Apricot Phase Four: Snowman++ and Reduced C-Chain Transaction Fees. https://medium.com/avalancheavax/apricot-phase-four-snowman-and-reduced-c-chain-transaction-fees-1e1f67b42ecf (2021)
4. Bartoletti, M., Chiang, J.H.y., Lluch-Lafuente, A.: A theory of Automated Market Makers in DeFi. In: International Conference on Coordination Languages and Models. pp. 168–187. Springer (2021), https://doi.org/10.1007/978-3-030-78142-2_11
5. Bartoletti, M., Chiang, J.H.y., Lluch-Lafuente, A.: Maximizing Extractable Value from Automated Market Makers. arXiv preprint arXiv:2106.01870 (2021), to appear in FC'22. https://arxiv.org/pdf/2106.01870
6. Baum, C., David, B., Dowsley, R.: Insured MPC: Efficient secure computation with financial penalties. In: International Conference on Financial Cryptography and Data Security. pp. 404–420. Springer (2020). https://doi.org/10.1007/978-3-030-51280-4_22
7. Baum, C., David, B., Frederiksen, T.K.: P2DEX: privacy-preserving decentralized cryptocurrency exchange. In: International Conference on Applied Cryptography and Network Security. pp. 163–194. Springer (2021). https://doi.org/10.1007/978-3-030-78372-3_7
8. Beerliova-Trubiniova, Z., Hirt, M.: Efficient multi-party computation with dispute control. In: Theory of Cryptography Conference. pp. 305–328. Springer (2006). https://doi.org/10.1007/11681878_16
9. Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E., Virza, M.: SNARKs for C: Verifying program executions succinctly and in zero knowledge. In: Annual cryptology conference. pp. 90–108. Springer (2013). https://doi.org/10.1007/978-3-642-40084-1_6
10. Bendlin, R., Damgård, I., Orlandi, C., Zakarias, S.: Semi-homomorphic encryption and multiparty computation. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 169–188. Springer, Heidelberg, Germany, Tallinn, Estonia (May 15–19, 2011). https://doi.org/10.1007/978-3-642-20465-4_11
11. Boneh, D., Naor, M.: Timed commitments. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 236–254. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 20–24, 2000). https://doi.org/10.1007/3-540-44598-6_15
12. Breidenbach, L., Daian, P., Tramèr, F., Juels, A.: Enter the Hydra: Towards Principled Bug Bounties and Exploit-Resistant Smart Contracts. In: 27th USENIX Security Symposium (USENIX Security 18). pp. 1335–1352. USENIX Association, Baltimore, MD (Aug 2018), https://www.usenix.org/conference/usenixsecurity18/presentation/breindenbach
13. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy (SP). pp. 315–334. IEEE (2018). https://doi.org/10.1109/SP.2018.00020

14. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: 34th ACM STOC. pp. 494–503. ACM Press, Montréal, Québec, Canada (May 19–21, 2002). https://doi.org/10.1145/509907.509980
15. Chitra, T., Angeris, G., Evans, A.: Differential Privacy in Constant Function Market Makers. Cryptology ePrint Archive (2021), https://eprint.iacr.org/2021/1101
16. Chu, S., Xia, Y., Zhang, Z.: Manta: a Plug and Play Private DeFi Stack (2021), https://eprint.iacr.org/2021/743
17. Daian, P., Goldfeder, S., Kell, T., Li, Y., Zhao, X., Bentov, I., Breidenbach, L., Juels, A.: Flash Boys 2.0: Frontrunning in Decentralized Exchanges, Miner Extractable Value, and Consensus Instability. In: IEEE Symposium on Security and Privacy. pp. 910–927. IEEE (2020). https://doi.org/10.1109/SP40000.2020.00040
18. Damgård, I., Nielsen, J.B.: Scalable and unconditionally secure multiparty computation. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 572–590. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2007). https://doi.org/10.1007/978-3-540-74143-5_32
19. Damgård, I., Pastro, V., Smart, N.P., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 643–662. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2012). https://doi.org/10.1007/978-3-642-32009-5_38
20. Danos, V., Khalloufi, H.E., Prat, J.: Global Order Routing on Exchange Networks. In: International Conference on Financial Cryptography and Data Security. pp. 207–226. Springer (2021). https://doi.org/10.1007/978-3-662-63958-0_19
21. Desmedt, Y., Frankel, Y.: Threshold cryptosystems. In: Brassard, G. (ed.) CRYPTO'89. LNCS, vol. 435, pp. 307–315. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 20–24, 1990). https://doi.org/10.1007/0-387-34805-0_28
22. Doweck, Y., Eyal, I.: Multi-party timed commitments. arXiv preprint arXiv:2005.04883 (2020), https://arxiv.org/pdf/2005.04883
23. Eskandari, S., Moosavi, S., Clark, J.: SoK: Transparent Dishonesty: Front-Running Attacks on Blockchain. In: Financial Cryptography. pp. 170–189. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-43725-1_13
24. da Gama, M.B., Cartlidge, J., Polychroniadou, A., Smart, N.P., Alaoui, Y.T.: Kicking-the-Bucket: Fast Privacy-Preserving Trading Using Buckets. Cryptology ePrint Archive, Report 2021/1549 (2021), to appear in FC'22, https://ia.cr/2021/1549
25. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct NIZKs without PCPs. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 626–645. Springer (2013). https://doi.org/10.1007/978-3-642-38348-9_37
26. Gentry, C.: Fully Homomorphic Encryption Using Ideal Lattices. In: Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing. p. 169–178. STOC '09, Association for Computing Machinery, New York, NY, USA (2009). https://doi.org/10.1145/1536414.1536440, https://doi.org/10.1145/1536414.1536440
27. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: Aho, A. (ed.) 19th ACM STOC. pp. 218–229. ACM Press, New York City, NY, USA (May 25–27, 1987). https://doi.org/10.1145/28395.28420

28. Groth, J.: Short pairing-based non-interactive zero-knowledge arguments. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 321–340. Springer, Heidelberg, Germany, Singapore (Dec 5–9, 2010). https://doi.org/10.1007/978-3-642-17373-8_19
29. Groth, J.: On the size of pairing-based non-interactive arguments. In: Annual international conference on the theory and applications of cryptographic techniques. pp. 305–326. Springer (2016). https://doi.org/10.1007/978-3-662-49896-5_11
30. Kelkar, M., Deb, S., Kannan, S.: Order-Fair Consensus in the Permissionless Setting. IACR Cryptol. ePrint Arch. **2021**, 139 (2021), https://eprint.iacr.org/2021/139
31. Kelkar, M., Deb, S., Long, S., Juels, A., Kannan, S.: Themis: Fast, Strong Order-Fairness in Byzantine Consensus. Cryptology ePrint Archive (2021), https://eprint.iacr.org/2021/1465
32. Kiayias, A., Zhou, H.S., Zikas, V.: Fair and robust multi-party computation using a global transaction ledger. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 705–734. Springer (2016). https://doi.org/10.1007/978-3-662-49896-5_25
33. Kursawe, K.: Wendy, the good little fairness widget: Achieving order fairness for blockchains. In: Proceedings of the 2nd ACM Conference on Advances in Financial Technologies. pp. 25–36 (2020). https://doi.org/10.1145/3419614.3423263
34. Li, Y.: HoneyBadgerSwap: Making MPC as a Sidechain. https://medium.com/initc3org/honeybadgerswap-making-mpc-as-a-sidechain-364bebdb10a5 (2021)
35. Lu, D., Yurek, T., Kulshreshtha, S., Govind, R., Kate, A., Miller, A.: Honeybadgermpc and asynchromix: Practical asynchronous mpc and its application to anonymous communication. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. pp. 887–903 (2019). https://doi.org/10.1145/3319535.3354238
36. Maxwell, G.: Confidential transactions. https://people.xiph.org/greg/confidential_values.txt, (2016)
37. Paradigm: Ethereum is a Dark Forest. https://www.paradigm.xyz/2020/08/ethereum-is-a-dark-forest/ (2020)
38. Parno, B., Howell, J., Gentry, C., Raykova, M.: Pinocchio: Nearly practical verifiable computation. In: 2013 IEEE Symposium on Security and Privacy. pp. 238–252. IEEE (2013). https://doi.org/10.1109/SP.2013.47
39. Penumbra: ZSwap documentation. https://protocol.penumbra.zone/main/zswap.html (2021)
40. Perez, D., Werner, S.M., Xu, J., Livshits, B.: Liquidations: DeFi on a Knife-edge. In: International Conference on Financial Cryptography and Data Security. pp. 457–476. Springer (2021). https://doi.org/10.1007/978-3-662-64331-0_24
41. Rivest, R.L., Shamir, A., Wagner, D.A.: Time-locked Puzzles and Time-release Crypto. https://people.csail.mit.edu/rivest/pubs/RSW96.pdf (1996)
42. Sasson, E.B., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy. pp. 459–474. IEEE (2014). https://doi.org/10.1109/SP.2014.36
43. Wang, Y., Chen, Y., Deng, S., Wattenhofer, R.: Cyclic Arbitrage in Decentralized Exchange Markets. Available at SSRN 3834535 (2021), https://dx.doi.org/10.2139/ssrn.3834535
44. White, D., Robinson, D., Adams, H.: Time-weighted Average Market Maker (TWAMM) (2021), https://www.paradigm.xyz/2021/07/twamm/

45. Yao, A.C.C.: Theory and applications of trapdoor functions (extended abstract). In: 23rd FOCS. pp. 80–91. IEEE Computer Society Press, Chicago, Illinois (Nov 3–5, 1982). https://doi.org/10.1109/SFCS.1982.45
46. Zhou, L., Qin, K., Cully, A., Livshits, B., Gervais, A.: On the just-in-time discovery of profit-generating transactions in defi protocols. arXiv preprint arXiv:2103.02228 (2021), https://arxiv.org/abs/2103.02228

## A Formalization: speculative sandwich

We formalize the example attack trace introduced in Section 3.2 (figs. 6 and 7) and prove that the attack strategy is either profitable or cost-neutral for the attacker. Again, we assume unit value of $\tau_0, \tau_1$ to be equal, and the initial AMM reserve state to be $(\mathsf{r} : \tau_0, \mathsf{r} : \tau_1)$: in this state, there is no arbitrage opportunity to be exploited, simplifying our analysis. We omit both AMM and transaction fees.

The victim $\mathsf{A}$ swap direction is *left*, inferred by $\mathsf{M}$ from $\mathsf{A}$'s public balance of $\mathsf{v}_\mathsf{A}^\mathsf{init} : \tau_0$ ($\mathsf{A}$ holds no units of $\tau_1$). The attack strategy is as follows:

1. **Round r**: Front-run victim with $\mathsf{M} : \mathsf{SL}(\mathsf{v}_\mathsf{M}^\mathsf{f} : \tau_0, \mathsf{w}_\mathsf{M}^\mathsf{f} : \tau_1)$ such that
$$(\mathsf{r} + \mathsf{v}_\mathsf{M}^\mathsf{f}) \cdot (\mathsf{r} - \mathsf{w}_\mathsf{M}^\mathsf{f}) = \mathsf{r}^2 \tag{1}$$
2. **Round r + 1**: Back-run victim in opposing direction to reestablish initial AMM reserve ratio, or if attacker balance is insufficient, back-run with largest amount available to attacker $\mathsf{M}$.

We must show that this strategy is always profitable (when the victim swap direction can be inferred by the attacker). We note that there are several variables beyond the attackers control. The ordering of both front-run and victim swap in round $r$ is random. Thus the desired "front-run" ordering of the victim swap in round $r$ may not succeed (the sandwich is unsuccessful if the victim swap precedes attacker front-run swap). Furthermore, the victim swap parameters can be arbitrarily chosen, so that the victim swap may not be *enabled* or execute in a given sequence. Thus, we must exhaustively demonstrate the profitability of the attacker strategy for all possible cases:

1) Successful sandwich & enabled victim swap
2) Successful sandwich & disabled victim swap
3) Unsuccessful sandwich & enabled victim swap
4) Unsuccessful sandwich & disabled victim swap

**Case 1:** *(Successful sandwich & enabled victim swap)*: We illustrate the symbolic execution of the attack trace below in terms of initial balances, chosen swap parameters and exchanged amounts.

$$\textcircled{0}\ \mathsf{A}[\mathsf{v}_\mathsf{A}^\mathsf{init} : \tau_0] \mid \mathsf{M}[\mathsf{v}_\mathsf{M}^\mathsf{init} : \tau_0, \mathsf{w}_\mathsf{M}^\mathsf{init} : \tau_1] \mid (\mathsf{r} : \tau_0, \mathsf{r} : \tau_1)$$

Round $\mathsf{r}$

$$\xrightarrow{\mathsf{M}:\mathsf{SL}(\mathsf{v}_\mathsf{M}^\mathsf{f}:\tau_0,\mathsf{w}_\mathsf{M}^\mathsf{f}:\tau_1)} \textcircled{1}\ \mathsf{A}[\mathsf{v}_\mathsf{A}^\mathsf{init} : \tau_0] \mid \mathsf{M}[\mathsf{v}_\mathsf{M}^\mathsf{init} - \mathsf{v}_\mathsf{M}^\mathsf{f} : \tau_0, \mathsf{w}_\mathsf{M}^\mathsf{init} + \mathsf{w}_\mathsf{M}^\mathsf{f} : \tau_1] \mid (\mathsf{r} + \mathsf{v}_\mathsf{M}^\mathsf{f} : \tau_0, \mathsf{r} - \mathsf{w}_\mathsf{M}^\mathsf{f} : \tau_1)$$

$$\xrightarrow{\mathsf{A}:\mathsf{SL}(\mathsf{v}_\mathsf{A}:\tau_0,\mathsf{w}_\mathsf{A}:\tau_1)} \textcircled{2}\ \mathsf{A}[\mathsf{v}_\mathsf{A}^\mathsf{init} - \mathsf{v}_\mathsf{A} : \tau_0, \mathsf{w}'_\mathsf{A} : \tau_1] \mid \mathsf{M}[\mathsf{v}_\mathsf{M}^\mathsf{init} - \mathsf{v}_\mathsf{M}^\mathsf{f} : \tau_0, \mathsf{w}_\mathsf{M}^\mathsf{init} + \mathsf{w}_\mathsf{M}^\mathsf{f} : \tau_1] \mid$$
$$(\mathsf{r} + \mathsf{v}_\mathsf{M}^\mathsf{f} + \mathsf{v}_\mathsf{A} : \tau_0, \mathsf{r} - \mathsf{w}_\mathsf{M}^\mathsf{f} - \mathsf{w}'_\mathsf{A} : \tau_1)$$

Round $\mathsf{r} + 1$

$$\xrightarrow{\mathsf{M}:\mathsf{SR}(\mathsf{v}_\mathsf{M}^\mathsf{b}:\tau_0,\mathsf{w}_\mathsf{M}^\mathsf{b}:\tau_1)} \textcircled{3}\ \mathsf{A}[\mathsf{v}_\mathsf{A}^\mathsf{init} - \mathsf{v}_\mathsf{A} : \tau_0, \mathsf{w}'_\mathsf{A} : \tau_1] \mid \mathsf{M}[\mathsf{v}_\mathsf{M}^\mathsf{init} - \mathsf{v}_\mathsf{M}^\mathsf{f} + \mathsf{v}_\mathsf{M}^{\mathsf{b}\,\prime} : \tau_0, \mathsf{w}_\mathsf{M}^\mathsf{init} + \mathsf{w}_\mathsf{M}^\mathsf{f} - \mathsf{w}_\mathsf{M}^\mathsf{b} : \tau_1] \mid$$
$$(\mathsf{r} + \mathsf{v}_\mathsf{M}^\mathsf{f} + \mathsf{v}_\mathsf{A} - \mathsf{v}_\mathsf{M}^{\mathsf{b}\,\prime} : \tau_0, \mathsf{r} - \mathsf{w}_\mathsf{M}^\mathsf{f} - \mathsf{w}'_\mathsf{A} + \mathsf{w}_\mathsf{M}^\mathsf{b} : \tau_1)$$

We show that the attack is profitable. For $\tau_0$ and $\tau_1$ of equal unit value, the net change in *value* exchanged by $\mathsf{M}$ must be positive. Thus, we must prove

$$\text{profit}_{\mathsf{M}} = -\mathsf{v}^{\mathsf{f}}_{\mathsf{M}} + \mathsf{w}^{\mathsf{f}}_{\mathsf{M}} - \mathsf{w}^{\mathsf{b}}_{\mathsf{M}} + \mathsf{v}^{\mathsf{b}}_{\mathsf{M}}{}' > 0 \tag{2}$$

Note that the amounts exchanged in the front-run are equal to the front-run parameters $(\mathsf{v}^{\mathsf{f}}_{\mathsf{M}}, \mathsf{w}^{\mathsf{f}}_{\mathsf{M}})$, as they are chosen such that (1) holds. We consider the **sub-case (a)** in which the attacker $\mathsf{M}$ has sufficient balance to perform the back-run swap such that the AMM reserves are restored to the original state and the **sub-case (b)** in which the attacker initially has no balance of $\tau_1$ to perform the back-run: $\mathsf{w}^{\mathsf{init}}_{\mathsf{M}} = 0$. Here, the funds of $\tau_1$ required to execute the back-run are received entirely in the front-run execution.

For **sub-case (a)**, we rewrite (2) in terms of independently chosen parameters $\mathsf{v}^{\mathsf{f}}_{\mathsf{M}}$, $\mathsf{v}_{\mathsf{A}}$ (the attacker only knows the victim swap direction) and initial reserve amounts $\mathsf{r}$. The reserves of the AMM are restored to the initial state in final state ③: summing all step changes to the reserves across the sandwich execution yields

$$\begin{aligned} \mathsf{r} + \mathsf{v}^{\mathsf{f}}_{\mathsf{M}} + \mathsf{v}_{\mathsf{A}} - \mathsf{v}^{\mathsf{b}}_{\mathsf{M}}{}' &= \mathsf{r} \qquad & \mathsf{r} - \mathsf{w}^{\mathsf{f}}_{\mathsf{M}} - \mathsf{w}'_{\mathsf{A}} + \mathsf{w}^{\mathsf{b}}_{\mathsf{M}} &= \mathsf{r} \\ \mathsf{v}^{\mathsf{f}}_{\mathsf{M}} + \mathsf{v}_{\mathsf{A}} - \mathsf{v}^{\mathsf{b}}_{\mathsf{M}}{}' &= 0 \qquad & - \mathsf{w}^{\mathsf{f}}_{\mathsf{M}} - \mathsf{w}'_{\mathsf{A}} + \mathsf{w}^{\mathsf{b}}_{\mathsf{M}} &= 0 \end{aligned}$$

or

$$\mathsf{v}^{\mathsf{b}}_{\mathsf{M}}{}' = \mathsf{v}^{\mathsf{f}}_{\mathsf{M}} + \mathsf{v}_{\mathsf{A}} \qquad \mathsf{w}^{\mathsf{b}}_{\mathsf{M}} = \mathsf{w}^{\mathsf{f}}_{\mathsf{M}} + \mathsf{w}'_{\mathsf{A}}$$

Inserting RHS of equations above into our proof obligation (2) yields

$$\begin{aligned} \text{profit}_{\mathsf{M}} = -\cancel{\mathsf{v}^{\mathsf{f}}_{\mathsf{M}}} + \cancel{\mathsf{v}^{\mathsf{f}}_{\mathsf{M}}} + \mathsf{v}_{\mathsf{A}} + \cancel{\mathsf{w}^{\mathsf{f}}_{\mathsf{M}}} - \cancel{\mathsf{w}^{\mathsf{f}}_{\mathsf{M}}} - \mathsf{w}'_{\mathsf{A}} &>^{?} 0 \\ \mathsf{v}_{\mathsf{A}} - \mathsf{w}'_{\mathsf{A}} &>^{?} 0 \end{aligned} \tag{3}$$

To evaluate whether this inequality holds, we must solve for $\mathsf{w}'_{\mathsf{A}}$ in terms of $\mathsf{v}_{\mathsf{A}}$ and $\mathsf{v}^{\mathsf{f}}_{\mathsf{M}}$ chosen independently by the victim and adversary respectively. We exploit the constant reserve product invariant which holds for across the entire execution.

$$\begin{aligned} (\mathsf{r} + \mathsf{v}^{\mathsf{f}}_{\mathsf{M}}) \cdot (\mathsf{r} - \mathsf{w}^{\mathsf{f}}_{\mathsf{M}}) &= \mathsf{r}^2 \quad \text{(front-run swap)} \\ (\mathsf{r} + \mathsf{v}^{\mathsf{f}}_{\mathsf{M}} + \mathsf{v}_{\mathsf{A}}) \cdot (\mathsf{r} - \mathsf{w}^{\mathsf{f}}_{\mathsf{M}} - \mathsf{w}'_{\mathsf{A}}) &= \mathsf{r}^2 \quad \text{(victim swap)} \end{aligned}$$

We can derive $\mathsf{r} - \mathsf{w}^{\mathsf{f}}_{\mathsf{M}} = \frac{\mathsf{r}^2}{\mathsf{r} + \mathsf{v}^{\mathsf{f}}_{\mathsf{M}}}$ from the first equation, and substitute the RHS for $\mathsf{r} - \mathsf{w}^{\mathsf{f}}_{\mathsf{M}}$ in the second equation to obtain

$$(\mathsf{r} + \mathsf{v}^{\mathsf{f}}_{\mathsf{M}} + \mathsf{v}_{\mathsf{A}}) \cdot \left(\frac{\mathsf{r}^2}{\mathsf{r} + \mathsf{v}^{\mathsf{f}}_{\mathsf{M}}} - \mathsf{w}'_{\mathsf{A}}\right) = \mathsf{r}^2$$

Solving for $\mathsf{w}_\mathsf{A}'$ ...

$$\begin{aligned}\mathsf{w}_\mathsf{A}' &= \frac{\mathsf{r}^2}{\mathsf{r}+\mathsf{v}_\mathsf{M}^\mathsf{f}} - \frac{\mathsf{r}^2}{\mathsf{r}+\mathsf{v}_\mathsf{M}^\mathsf{f}+\mathsf{v}_\mathsf{A}} \\ &= \frac{\mathsf{r}^2(\mathsf{r}+\mathsf{v}_\mathsf{M}^\mathsf{f}+\mathsf{v}_\mathsf{A}) - \mathsf{r}^2(\mathsf{r}+\mathsf{v}_\mathsf{M}^\mathsf{f})}{(\mathsf{r}+\mathsf{v}_\mathsf{M}^\mathsf{f})(\mathsf{r}+\mathsf{v}_\mathsf{M}^\mathsf{f}+\mathsf{v}_\mathsf{A})} \\ &= \frac{\mathsf{r}^2}{\mathsf{r}^2+(2\mathsf{v}_\mathsf{M}^\mathsf{f}+\mathsf{v}_\mathsf{A})\mathsf{r}+(\mathsf{v}_\mathsf{M}^\mathsf{f})^2+\mathsf{v}_\mathsf{A}\mathsf{v}_\mathsf{M}^\mathsf{f}} \cdot \mathsf{v}_\mathsf{A}\end{aligned}$$

and substituting the RHS for $\mathsf{w}_\mathsf{A}'$ in the proof obligation in (3) finally yields

$$\text{profit}_\mathsf{M} = (1 - \frac{\mathsf{r}^2}{\mathsf{r}^2+(2\mathsf{v}_\mathsf{M}^\mathsf{f}+\mathsf{v}_\mathsf{A})\mathsf{r}+(\mathsf{v}_\mathsf{M}^\mathsf{f})^2+\mathsf{v}_\mathsf{A}\mathsf{v}_\mathsf{M}^\mathsf{f}}) \cdot \mathsf{v}_\mathsf{A} > 0 \tag{4}$$

The fraction expression above is less than 1 for any choice of positive $\mathsf{v}_\mathsf{M}^\mathsf{f}$ and $\mathsf{v}_\mathsf{A}$ as the numerator is smaller than the denominator. The attacker profit is thus positive and increases with $\mathsf{v}_\mathsf{M}$, justifying the front-run swap by $\mathsf{M}$.

Next, we consider the **sub-case (b)**, where the attacker initially has no balance of $\tau_1$, and restate the profit of attacker for the reader's convenience.

$$\text{profit}_\mathsf{M} = -\mathsf{v}_\mathsf{M}^\mathsf{f} + \mathsf{w}_\mathsf{M}^\mathsf{f} - \mathsf{w}_\mathsf{M}^\mathsf{b} + {\mathsf{v}_\mathsf{M}^\mathsf{b}}' >^? 0$$

We assume initial attacker balance in $\mathsf{w}_\mathsf{M}^\mathsf{init} : \tau_1$ to be $0 : \tau_1$, so that all the amount of $\tau_1$ available for the back-run in state ② is received in the front-run: thus, substituting $\mathsf{w}_\mathsf{M}^\mathsf{b} = \mathsf{w}_\mathsf{M}^\mathsf{f}$ into the equation above yields

$$\text{profit}_\mathsf{M} = -\mathsf{v}_\mathsf{M}^\mathsf{f} + {\mathsf{v}_\mathsf{M}^\mathsf{b}}' >^? 0 \tag{5}$$

To prove this inequality, we solve for ${\mathsf{v}_\mathsf{M}^\mathsf{b}}'$ in terms of $\mathsf{v}_\mathsf{M}^\mathsf{f}$ and $\mathsf{v}_\mathsf{A}$ chosen independently by the victim and adversary respectively and initial reserves amounts $\mathsf{r}$. We exploit the constant reserve product invariant which holds throughout the execution.

$$\begin{aligned}(\mathsf{r}+\mathsf{v}_\mathsf{M}^\mathsf{f}) \cdot (\mathsf{r}-\mathsf{w}_\mathsf{M}^\mathsf{f}) &= \mathsf{r}^2 \quad \text{(Front-run)} \\ (\mathsf{r}+\mathsf{v}_\mathsf{M}^\mathsf{f}+\mathsf{v}_\mathsf{A}) \cdot (\mathsf{r}-\mathsf{w}_\mathsf{M}^\mathsf{f}-\mathsf{w}_\mathsf{A}') &= \mathsf{r}^2 \quad \text{(Victim swap)} \\ (\mathsf{r}+\mathsf{v}_\mathsf{M}^\mathsf{f}+\mathsf{v}_\mathsf{A}-{\mathsf{v}_\mathsf{M}^\mathsf{b}}') \cdot (\mathsf{r}-\mathsf{w}_\mathsf{M}^\mathsf{f}-\mathsf{w}_\mathsf{A}'+\mathsf{w}_\mathsf{M}^\mathsf{b}) &= \mathsf{r}^2 \quad \text{(Back-run)}\end{aligned}$$

Since $\mathsf{w}_\mathsf{M}^\mathsf{f} = \mathsf{w}_\mathsf{M}^\mathsf{b}$ is assumed in sub-case (b), the 3rd equation (back-run) yields

$${\mathsf{v}_\mathsf{M}^\mathsf{b}}' = \mathsf{r} + \mathsf{v}_\mathsf{M}^\mathsf{f} + \mathsf{v}_\mathsf{A} - \frac{\mathsf{r}^2}{\mathsf{r}-\mathsf{w}_\mathsf{A}'} \tag{6}$$

From the 2nd equation (victim swap), we solve for $\mathsf{w}_\mathsf{A}'$ in terms of independent parameters $\mathsf{v}_\mathsf{M}^\mathsf{f}$, $\mathsf{v}_\mathsf{A}$ and $\mathsf{r}$

$$\mathsf{w}_\mathsf{A}' = \mathsf{r} - \mathsf{w}_\mathsf{M}^\mathsf{f} - \frac{\mathsf{r}^2}{\mathsf{r}+\mathsf{v}_\mathsf{M}^\mathsf{f}+\mathsf{v}_\mathsf{A}}$$

From the 1st equation (front-run) $\mathsf{w}^{\mathsf{f}}_{\mathsf{M}} = \frac{\mathsf{r} \cdot \mathsf{v}^{\mathsf{f}}_{\mathsf{M}}}{\mathsf{r} + \mathsf{v}^{\mathsf{f}}_{\mathsf{M}}}$, so we can rewrite the above as

$$\mathsf{w}'_{\mathsf{A}} = \mathsf{r} - \frac{\mathsf{r} \cdot \mathsf{v}^{\mathsf{f}}_{\mathsf{M}}}{\mathsf{r} + \mathsf{v}^{\mathsf{f}}_{\mathsf{M}}} - \frac{\mathsf{r}^2}{\mathsf{r} + \mathsf{v}^{\mathsf{f}}_{\mathsf{M}} + \mathsf{v}_{\mathsf{A}}} = \frac{\mathsf{r}^2}{\mathsf{r} + \mathsf{v}^{\mathsf{f}}_{\mathsf{M}}} - \frac{\mathsf{r}^2}{\mathsf{r} + \mathsf{v}^{\mathsf{f}}_{\mathsf{M}} + \mathsf{v}_{\mathsf{A}}} = \frac{\mathsf{r}^2 \cdot \mathsf{v}_{\mathsf{A}}}{(\mathsf{r} + \mathsf{v}^{\mathsf{f}}_{\mathsf{M}})(\mathsf{r} + \mathsf{v}^{\mathsf{f}}_{\mathsf{M}} + \mathsf{v}_{\mathsf{A}})}$$

$$\mathsf{r} - \mathsf{w}'_{\mathsf{A}} = \frac{\mathsf{r}(\mathsf{r} + \mathsf{v}^{\mathsf{f}}_{\mathsf{M}})(\mathsf{r} + \mathsf{v}^{\mathsf{f}}_{\mathsf{M}} + \mathsf{v}_{\mathsf{A}}) - \mathsf{r}^2 \cdot \mathsf{v}_{\mathsf{A}}}{(\mathsf{r} + \mathsf{v}^{\mathsf{f}}_{\mathsf{M}})(\mathsf{r} + \mathsf{v}^{\mathsf{f}}_{\mathsf{M}} + \mathsf{v}_{\mathsf{A}})}$$

Substituting the RHS above for $\mathsf{r} - \mathsf{w}'_{\mathsf{A}}$ in the denominator expression of (6) and then substituting the RHS of (6) for $\mathsf{v}^{\mathsf{b}}_{\mathsf{M}}{}'$ in (5) yields

$$\begin{aligned}
\text{profit}_{\mathsf{M}} &= -\cancel{\mathsf{v}^{\mathsf{f}}_{\mathsf{M}}} + \mathsf{r} + \cancel{\mathsf{v}^{\mathsf{f}}_{\mathsf{M}}} + \mathsf{v}_{\mathsf{A}} - \frac{\mathsf{r}^2(\mathsf{r} + \mathsf{v}^{\mathsf{f}}_{\mathsf{M}})(\mathsf{r} + \mathsf{v}^{\mathsf{f}}_{\mathsf{M}} + \mathsf{v}_{\mathsf{A}})}{\mathsf{r}(\mathsf{r} + \mathsf{v}^{\mathsf{f}}_{\mathsf{M}})(\mathsf{r} + \mathsf{v}^{\mathsf{f}}_{\mathsf{M}} + \mathsf{v}_{\mathsf{A}}) - \mathsf{r}^2 \cdot \mathsf{v}_{\mathsf{A}}} \\
&= \mathsf{v}_{\mathsf{A}} - \frac{\mathsf{r}^3 \mathsf{v}_{\mathsf{A}}}{\mathsf{r}(\mathsf{r} + \mathsf{v}^{\mathsf{f}}_{\mathsf{M}})(\mathsf{r} + \mathsf{v}^{\mathsf{f}}_{\mathsf{M}} + \mathsf{v}_{\mathsf{A}}) - \mathsf{r}^2 \cdot \mathsf{v}_{\mathsf{A}}} \\
&= (1 - \frac{\mathsf{r}^2 \mathsf{v}_{\mathsf{A}}}{(\mathsf{r} + \mathsf{v}^{\mathsf{f}}_{\mathsf{M}})(\mathsf{r} + \mathsf{v}^{\mathsf{f}}_{\mathsf{M}} + \mathsf{v}_{\mathsf{A}}) - \mathsf{r} \cdot \mathsf{v}_{\mathsf{A}}}) \cdot \mathsf{v}_{\mathsf{A}} \\
&= (1 - \frac{\mathsf{r}^2}{\mathsf{r}^2 + 2\mathsf{v}^{\mathsf{f}}_{\mathsf{M}}\mathsf{r} + (\mathsf{v}^{\mathsf{f}}_{\mathsf{M}})^2 + \mathsf{v}_{\mathsf{A}}\mathsf{v}^{\mathsf{f}}_{\mathsf{M}}}) \cdot \mathsf{v}_{\mathsf{A}} \qquad (7)
\end{aligned}$$

The attacker profit is positive but strictly less than the gain (4) obtained in sub-case (a).

**Case 2** *(Successful sandwich & disabled victim swap)*: Should the victim swap not execute in round $r$, then $\mathsf{M}$ can simply revert the state of the AMM with a back-run in the round $r + 1$ with the same parameter values as in the front-run.

$$⓪\ \mathsf{A}[\mathsf{v}^{\mathsf{init}}_{\mathsf{A}} : \tau_0] \mid \mathsf{M}[\mathsf{v}^{\mathsf{init}}_{\mathsf{A}} : \tau_0, \mathsf{w}^{\mathsf{init}}_{\mathsf{A}} : \tau_1] \mid (\mathsf{r} : \tau_0, \mathsf{r} : \tau_1)$$

Round $\mathsf{r}$

$$\xrightarrow{\mathsf{M}:\mathsf{SL}(\mathsf{v}^{\mathsf{f}}_{\mathsf{M}}:\tau_0,\mathsf{w}^{\mathsf{f}}_{\mathsf{M}}:\tau_1)} ①\ \mathsf{A}[\mathsf{v}^{\mathsf{init}}_{\mathsf{A}} : \tau_0] \mid \mathsf{M}[\mathsf{v}^{\mathsf{init}}_{\mathsf{M}} - \mathsf{v}^{\mathsf{f}}_{\mathsf{M}} : \tau_0, \mathsf{w}^{\mathsf{init}}_{\mathsf{M}} + \mathsf{w}^{\mathsf{f}}_{\mathsf{M}} : \tau_1] \mid (\mathsf{r} + \mathsf{v}^{\mathsf{f}}_{\mathsf{M}} : \tau_0, \mathsf{r} - \mathsf{w}^{\mathsf{f}}_{\mathsf{M}} : \tau_1)$$

$$\xrightarrow{\cancel{\mathsf{A}:\mathsf{SL}(\mathsf{v}_{\mathsf{A}}:\tau_0,\mathsf{w}_{\mathsf{A}}:\tau_1)}} ②\ \mathsf{A}[\mathsf{v}^{\mathsf{init}}_{\mathsf{A}} : \tau_0] \mid \mathsf{M}[\mathsf{v}^{\mathsf{init}}_{\mathsf{M}} - \mathsf{v}^{\mathsf{f}}_{\mathsf{M}} : \tau_0, \mathsf{w}^{\mathsf{init}}_{\mathsf{M}} + \mathsf{w}^{\mathsf{f}}_{\mathsf{M}} : \tau_1] \mid (\mathsf{r} + \mathsf{v}^{\mathsf{f}}_{\mathsf{M}} : \tau_0, \mathsf{r} - \mathsf{w}^{\mathsf{f}}_{\mathsf{M}} : \tau_1)$$

Round $\mathsf{r} + 1$

$$\xrightarrow{\mathsf{M}:\mathsf{SR}(\mathsf{v}^{\mathsf{f}}_{\mathsf{M}}:\tau_0,\mathsf{w}^{\mathsf{f}}_{\mathsf{M}}:\tau_1)} ③\ \mathsf{A}[\mathsf{v}^{\mathsf{init}}_{\mathsf{A}} : \tau_0] \mid \mathsf{M}[\mathsf{v}^{\mathsf{init}}_{\mathsf{M}} : \tau_0, \mathsf{w}^{\mathsf{init}}_{\mathsf{M}} : \tau_1] \mid (\mathsf{r} : \tau_0, \mathsf{r} : \tau_1)$$

The attack execution is trivially cost-neutral for $\mathsf{M}$.

**Case 3** *(Failed sandwich & enabled victim swap)*: We must show that the attacker front-run must be disabled assuming the attacker parameters are chosen as described in the attack strategy. Further, we can demonstrate that the back-run by the attacker is profitable.

$$\textcircled{0}\ \mathsf{A}[\mathsf{v}_\mathsf{A}^{\mathsf{init}} : \tau_0] \mid \mathsf{M}[\mathsf{v}_\mathsf{M}^{\mathsf{init}} : \tau_0, \mathsf{w}_\mathsf{M}^{\mathsf{init}} : \tau_1] \mid (\mathsf{r} : \tau_0, \mathsf{r} : \tau_1)$$

Round r

$$\xrightarrow{\mathsf{A}:\mathsf{SL}(\mathsf{v}_\mathsf{A}:\tau_0,\mathsf{w}_\mathsf{A}:\tau_1)} \textcircled{1}\ \mathsf{A}[\mathsf{v}_\mathsf{A}^{\mathsf{init}} - \mathsf{v}_\mathsf{A} : \tau_0, \mathsf{w}'_\mathsf{A} : \tau_1] \mid \mathsf{M}[\mathsf{v}_\mathsf{M}^{\mathsf{init}} : \tau_0, \mathsf{w}_\mathsf{M}^{\mathsf{init}} : \tau_1] \mid (\mathsf{r} + \mathsf{v}_\mathsf{A} : \tau_0, \mathsf{r} - \mathsf{w}'_\mathsf{A} : \tau_1)$$

$$\xrightarrow{\mathsf{M}:\cancel{\mathsf{SL}(\mathsf{v}_\mathsf{M}^\mathsf{f}:\tau_0,\mathsf{w}_\mathsf{M}^\mathsf{f}:\tau_1)}} \textcircled{2}\ \mathsf{A}[\mathsf{v}_\mathsf{A}^{\mathsf{init}} - \mathsf{v}_\mathsf{A} : \tau_0, \mathsf{w}'_\mathsf{A} : \tau_1] \mid \mathsf{M}[\mathsf{v}_\mathsf{M}^{\mathsf{init}} : \tau_0, \mathsf{w}_\mathsf{M}^{\mathsf{init}} : \tau_1] \mid (\mathsf{r} + \mathsf{v}_\mathsf{A} : \tau_0, \mathsf{r} - \mathsf{w}'_\mathsf{A} : \tau_1)$$

Round r + 1

$$\xrightarrow{\mathsf{M}:\mathsf{SR}(\mathsf{v}_\mathsf{M}^\mathsf{b}:\tau_0,\mathsf{w}_\mathsf{M}^\mathsf{b}:\tau_1)} \textcircled{3}\ \mathsf{A}[\mathsf{v}_\mathsf{A}^{\mathsf{init}} - \mathsf{v}_\mathsf{A} : \tau_0, \mathsf{w}'_\mathsf{A} : \tau_1] \mid \mathsf{M}[\mathsf{v}_\mathsf{M}^{\mathsf{init}} + \mathsf{v}_\mathsf{M}^{\mathsf{b}\,\prime} : \tau_0, \mathsf{w}_\mathsf{M}^{\mathsf{init}} - \mathsf{w}_\mathsf{M}^\mathsf{b} : \tau_1] \mid (\mathsf{r} : \tau_0, \mathsf{r} : \tau_1)$$

As described in step (1) of attack strategy, M's front-run parameters are chosen such that

$$(\mathsf{r} + \mathsf{v}_\mathsf{M}^\mathsf{f}) \cdot (\mathsf{r} - \mathsf{w}_\mathsf{M}^\mathsf{f}) = \mathsf{r}^2$$
$$\mathsf{w}_\mathsf{M}^\mathsf{f} = \frac{\mathsf{r} \cdot \mathsf{v}_\mathsf{M}^\mathsf{f}}{\mathsf{r} + \mathsf{v}_\mathsf{M}^\mathsf{f}} \tag{8}$$

Thus, the front-run swap is only enabled if the received amount is equal or greater to $\mathsf{w}_\mathsf{M}^\mathsf{f}$ shown above. Note, that this doesn't hold if the front-run is executed in state ① of case (3) following the enabled victim swap. We prove this by contradiction: assume that the front-run executes following the victim swap, then the constant reserve product invariant must hold.

$$(\mathsf{r} + \mathsf{v}_\mathsf{A}) \cdot (\mathsf{r} - \mathsf{w}'_\mathsf{A}) = \mathsf{r}^2 \quad \text{(Victim swap)}$$
$$(\mathsf{r} + \mathsf{v}_\mathsf{A} + \mathsf{v}_\mathsf{M}^\mathsf{f}) \cdot (\mathsf{r} - \mathsf{w}'_\mathsf{A} - \mathsf{w}_\mathsf{M}^{\mathsf{f}\,\prime}) = \mathsf{r}^2 \quad \text{(Front-run)}$$

We solve for $(\mathsf{r} - \mathsf{w}'_\mathsf{A})$ in the first equation and insert into the second equation to obtain

$$(\mathsf{r} + \mathsf{v}_\mathsf{A} + \mathsf{v}_\mathsf{M}^\mathsf{f}) \cdot (\frac{\mathsf{r}^2}{\mathsf{r} + \mathsf{v}_\mathsf{A}} - \mathsf{w}_\mathsf{M}^{\mathsf{f}\,\prime}) = \mathsf{r}^2$$

Further, we solve for $\mathsf{w}_\mathsf{M}^{\mathsf{f}\,\prime}$ in terms of $\mathsf{r}$, $\mathsf{v}_\mathsf{A}$ and $\mathsf{v}_\mathsf{M}^\mathsf{f}$

$$\frac{\mathsf{r}^2}{\mathsf{r} + \mathsf{v}_\mathsf{A}} - \mathsf{w}_\mathsf{M}^{\mathsf{f}\,\prime} = \frac{\mathsf{r}^2}{(\mathsf{r} + \mathsf{v}_\mathsf{A} + \mathsf{v}_\mathsf{M}^\mathsf{f})}$$

$$\mathsf{w}_\mathsf{M}^{\mathsf{f}\,\prime} = \frac{\mathsf{r}^2}{\mathsf{r} + \mathsf{v}_\mathsf{A}} - \frac{\mathsf{r}^2}{\mathsf{r} + \mathsf{v}_\mathsf{A} + \mathsf{v}_\mathsf{M}^\mathsf{f}} = \frac{\mathsf{r}^2 \cdot \mathsf{v}_\mathsf{M}^\mathsf{f}}{(\mathsf{r} + \mathsf{v}_\mathsf{A}) \cdot (\mathsf{r} + \mathsf{v}_\mathsf{A} + \mathsf{v}_\mathsf{M}^\mathsf{f})} = \frac{\mathsf{r}}{\mathsf{r} + \mathsf{v}_\mathsf{A}} \cdot \frac{\mathsf{r} \cdot \mathsf{v}_\mathsf{M}^\mathsf{f}}{(\mathsf{r} + \mathsf{v}_\mathsf{A} + \mathsf{v}_\mathsf{M}^\mathsf{f})}$$

Comparing with $\mathsf{w}_\mathsf{M}^\mathsf{f}$ in (8), we can infer the following inequality

$$\mathsf{w}_\mathsf{M}^{\mathsf{f}\,\prime} < \mathsf{w}_\mathsf{M}^\mathsf{f}$$

which cannot hold in a valid execution by definition of swaps: a user cannot receive less than the chosen swap limit. Thus, the front-run cannot be enabled in state ① of case (3).

Next, we prove the profitability of the back-run. Assuming a sufficient balance of the attacker to revert the effect of the victim swap, the swap parameters of the back-run can be chosen to reverse the affects of victim swap on the AMM reserves, which $\mathsf{M}$ observes following the output-phase of round $r$: namely, $\mathsf{v}_{\mathsf{M}}^{\mathsf{b}} = \mathsf{v}_{\mathsf{A}}$ and $\mathsf{w}_{\mathsf{M}}^{\mathsf{b}} = \mathsf{w}_{\mathsf{A}}'$. We insert these into the reserve product invariant from the victim swap

$$(\mathsf{r} + \mathsf{v}_{\mathsf{A}}) \cdot (\mathsf{r} - \mathsf{w}_{\mathsf{A}}') = \mathsf{r}^2 \quad \text{(Victim swap)}$$

to obtain

$$\begin{aligned}(\mathsf{r} + \mathsf{v}_{\mathsf{M}}^{\mathsf{b}}) \cdot (\mathsf{r} - \mathsf{w}_{\mathsf{M}}^{\mathsf{b}}) = \mathsf{r}^2 \\ \mathsf{w}_{\mathsf{M}}^{\mathsf{b}} &= \frac{\mathsf{r}}{\mathsf{r} + \mathsf{v}_{\mathsf{M}}^{\mathsf{b}}} \cdot \mathsf{v}_{\mathsf{M}}^{\mathsf{b}} \\ \mathsf{w}_{\mathsf{M}}^{\mathsf{b}} &< \mathsf{v}_{\mathsf{M}}^{\mathsf{b}}\end{aligned}$$

For equal unit value of both token types, this is clearly profitable, as $\mathsf{M}$ receives more value ($\mathsf{v}_{\mathsf{M}}^{\mathsf{b}}$) as it sends ($\mathsf{w}_{\mathsf{M}}^{\mathsf{b}}$). If attacker has no balance of $\tau_1$ it simply omits the back-run and the attack is aborted, resulting in a cost-neutral execution for the attacker.

**Case 4** *(Failed sandwich & disabled victim swap)*: As in case (2) - should the victim swap not execute in round $\mathsf{r}$, then $\mathsf{M}$ can simply revert the state of the AMM with a back-run in the round $\mathsf{r} + 1$

$$\textcircled{0}\ \mathsf{A}[\mathsf{v}_{\mathsf{A}}^{\mathsf{init}} : \tau_0] \mid \mathsf{M}[\mathsf{v}_{\mathsf{A}}^{\mathsf{init}} : \tau_0, \mathsf{w}_{\mathsf{A}}^{\mathsf{init}} : \tau_1] \mid (\mathsf{r} : \tau_0, \mathsf{r} : \tau_1)$$

Round $\mathsf{r}$

$$\xrightarrow{\cancel{\mathsf{A}:\mathsf{SL}(\mathsf{v}_{\mathsf{A}}:\tau_0,\mathsf{w}_{\mathsf{A}}:\tau_1)}} \textcircled{1}\ \mathsf{A}[\mathsf{v}_{\mathsf{A}}^{\mathsf{init}} : \tau_0] \mid \mathsf{M}[\mathsf{v}_{\mathsf{M}}^{\mathsf{init}} : \tau_0, \mathsf{w}_{\mathsf{M}}^{\mathsf{init}} : \tau_1] \mid (\mathsf{r} : \tau_0, \mathsf{r} : \tau_1)$$

$$\xrightarrow{\mathsf{M}:\mathsf{SL}(\mathsf{v}_{\mathsf{M}}^{\mathsf{f}}:\tau_0,\mathsf{w}_{\mathsf{M}}^{\mathsf{f}}:\tau_1)} \textcircled{2}\ \mathsf{A}[\mathsf{v}_{\mathsf{A}}^{\mathsf{init}} : \tau_0] \mid \mathsf{M}[\mathsf{v}_{\mathsf{M}}^{\mathsf{init}} - \mathsf{v}_{\mathsf{M}}^{\mathsf{f}} : \tau_0, \mathsf{w}_{\mathsf{M}}^{\mathsf{init}} + \mathsf{w}_{\mathsf{M}}^{\mathsf{f}} : \tau_1] \mid (\mathsf{r} + \mathsf{v}_{\mathsf{M}}^{\mathsf{f}} : \tau_0, \mathsf{r} - \mathsf{w}_{\mathsf{M}}^{\mathsf{f}} : \tau_1)$$

Round $\mathsf{r} + 1$

$$\xrightarrow{\mathsf{M}:\mathsf{SR}(\mathsf{v}_{\mathsf{M}}^{\mathsf{f}}:\tau_0,\mathsf{w}_{\mathsf{M}}^{\mathsf{f}}:\tau_1)} \textcircled{3}\ \mathsf{A}[\mathsf{v}_{\mathsf{A}}^{\mathsf{init}} : \tau_0] \mid \mathsf{M}[\mathsf{v}_{\mathsf{M}}^{\mathsf{init}} : \tau_0, \mathsf{w}_{\mathsf{M}}^{\mathsf{init}} : \tau_1] \mid (\mathsf{r} : \tau_0, \mathsf{r} : \tau_1)$$

The attack execution is trivially cost-neutral for $\mathsf{M}$.

## B Speculative sandwich with private user balances

Importantly, when performing the speculative AMM swap attack shown in Figures 6 and 7 and formalized in Appendix A, the direction of the victim swap must be known. If user balances are private, $\mathsf{M}$ will have to guess the direction of the front-running swap. However, this is not a profitable strategy: an incorrect guess can result in a loss for $\mathsf{M}$ as shown in the trivial example execution below.

$$\begin{array}{c}
\mathsf{A}[10:\tau_0,10:\tau_1] \mid \mathsf{M}[7:\tau_0,15:\tau_1] \mid (100:\tau_0,100:\tau_1) \\
\hline
\text{Round } \mathsf{r} \\
\xrightarrow{\mathsf{M}:\mathsf{SL}(7:\tau_0,6.5:\tau_1)} \mathsf{A}[10:\tau_0,10:\tau_1] \mid \mathsf{M}[21.5:\tau_1] \mid (107:\tau_0,93.5:\tau_1) \\
\xrightarrow{\mathsf{A}:\mathsf{SR}(7:\tau_0,6.5:\tau_1)} \mathsf{A}[17:\tau_0,3.5:\tau_1] \mid \mathsf{M}[21.5:\tau_1] \mid (100:\tau_0,100:\tau_1)
\end{array}$$

Again, assuming equal unit value of $\tau_0$ and $\tau_1$, $\mathsf{M}$ realizes a loss of $7+15-21.5 = 0.5$. No back-run swap is possible that extracts any arbitrage value given that the reserve ratio is already consistent with the assumption that unit values of $\tau_0$ and $\tau_1$ are equal [4]. Thus, speculative sandwich attacks are only rational if the victim swap direction can be inferred, motivating the need for private user balances.

## C Example: speculative sandwich of scheduled swap

We illustrate an example of a sandwich of a scheduled swap. Such an attack can be exploited despite the batching of blinded user inputs §3.2, as long as input schedules remain public [44]. Let $\mathsf{A} : \mathsf{SL}(20 : \tau_0, 15 : \tau_1, \mathsf{r})$ be a swap action that is scheduled to execute as soon as possible *following* block-chain round $\mathsf{r}$, thus requiring no further interaction from the user. Further, let the set of scheduled swap orders be captured in a publicly observable state fragment, i.e. $\Gamma = [\,\mathsf{A} : \mathsf{SL}(15 : \tau_0, 10 : \tau_1, \mathsf{r})\,]$. In practice, such a scheduled swap order will be *evaluated* prior to the first swap order in round $\mathsf{r}+1$, so that it is not possible for the adversary to place a front-run swap before it in round $\mathsf{r}+1$.

However, the sandwich attack can still be executed by an adversary which prevents honest users from submitting swap. The adversary simply submits the front-run to round $\mathsf{r}$, and the back-run to round $\mathsf{r}+1$, whilst suppressing all other user inputs.

$$\begin{array}{c}
\mathsf{A}[15:\tau_0] \mid \mathsf{M}[15:\tau_0,10:\tau_1] \mid (100:\tau_0,100:\tau_1) \mid \Gamma \\
\hline
\text{Round } \mathsf{r} \\
\xrightarrow{\mathsf{M}:\mathsf{SL}(15:\tau_0,13:\tau_1)} \mathsf{A}[15:\tau_0] \mid \mathsf{M}[23:\tau_1] \mid (115:\tau_0,87:\tau_1) \mid \Gamma \\
\hline
\text{Round } \mathsf{r}+1 \\
\xrightarrow{\mathsf{A}:\mathsf{SL}(15:\tau_0,10:\tau_1,\mathsf{r})} \mathsf{A}[10:\tau_1] \mid \mathsf{M}[23:\tau_1] \mid (130:\tau_0,77:\tau_1) \mid \\
\Gamma \setminus [\,\mathsf{A}:\mathsf{SL}(15:\tau_0,10:\tau_1),\mathsf{r}\,] \\
\xrightarrow{\mathsf{M}:\mathsf{SR}(30:\tau_0,23:\tau_1)} \mathsf{A}[10:\tau_1] \mid \mathsf{M}[30:\tau_0] \mid (100:\tau_0,100:\tau_1) \mid \\
\Gamma \setminus [\,\mathsf{A}:\mathsf{SL}(15:\tau_0,10:\tau_1,\mathsf{r})\,]
\end{array}$$

We emphasize that scheduled swap orders do not require the submitting user $\mathsf{A}$ to participate in the round it is scheduled: it is evaluated automatically by the application. Furthermore, since the victim's swap parameters are public, the front-run and back-run parameters can be chosen to optimize M's profit.

## D Speculative sandwich in hash-based commit & reveal schemes

As shown in Appendix A, the speculative sandwich attack is rational as long as the direction of the victim swap is known. Hash-based commit & reveal schemes suffer from selective output by the adversary (Figure 4), permitting a speculative attack to succeed even if the swap direction cannot be inferred from public user balances. Here the attacker simply commits two front-run swaps of opposing directions in the same round as the victim swap, whilst suppressing other user inputs. In the output-phase, the adversary learns the direction of the victim swap before having to open its own commitments and selectively opens the front-run of the same direction as the victim swap, whilst refraining from opening the other front-run swap. The back-run is then executed as in Appendix A.