

A quantitative notion of economic security for smart contract compositions

Emily Priyadarshini¹ and Massimo Bartoletti²

¹ IISER Pune, India

² University of Cagliari, Italy

Abstract. Decentralized applications are often composed of multiple interconnected smart contracts. This is especially evident in DeFi, where protocols are heavily intertwined and rely on a variety of basic building blocks such as tokens, decentralized exchanges and lending protocols. A crucial security challenge in this setting arises when adversaries target individual components to cause systemic economic losses. Existing security notions focus on determining the existence of these attacks, but fail to quantify the effect of manipulating individual components on the overall economic security of the system. In this paper, we introduce a quantitative security notion that measures how an attack on a single component can amplify economic losses of the overall system. We study the fundamental properties of this notion and apply it to assess the security of key compositions. In particular, we analyse under-collateralized loan attacks in systems made of lending protocols and decentralized exchanges.

1 Introduction

Developing decentralized applications nowadays involves suitably designing, assembling and customizing a multitude of smart contracts, resulting in complex interactions and dependencies. In particular, recent DeFi applications are highly interconnected compositions of smart contracts of various kinds, including tokens, derivatives, decentralized exchanges (DEX), and lending protocols [15,16].

This complexity poses significant security risks, as adversaries targeting one of the components may compromise the security of the overall application. Note that, for this to happen, the attacked component does not even need to have a proper vulnerability to exploit. For example, in an application composed of a lending protocol and a DEX serving as a price oracle, adversaries could target the DEX in order to artificially inflate the price of an asset that they have previously deposited to the lending pool. This manipulation would allow adversaries to borrow other assets with an insufficient collateral, circumventing the intended economic mechanism of the lending protocol [12,21,5,20,1].

The first step to address these risks is to formally define when a system of smart contracts is secure. In recent years, a few security notions have emerged, starting from Babel, Daian, Kelkar and Juels' "Clockwork finance" [3]. Broadly, these definitions try to characterise the economic security of smart contract systems based on the extent of economic damage that adversaries can inflict

on them. In this context, adversaries are typically assumed to have the powers of consensus nodes. Namely, they can reorder, drop or insert transactions in blocks. Accordingly, the economic damage on a system S can be quantified in terms of the Maximal Extractable Value (MEV) that adversaries can extract from S by leveraging these powers [10]. To provide a more concrete formulation of the existing notions, consider a set of contracts Δ to be deployed in a system S . We denote by $S \mid \Delta$ the system composed of S and Δ . The security criterion in [3] requires that $\text{MEV}(S \mid \Delta) \leq (1+\varepsilon) \text{MEV}(S)$: namely, the MEV extractable from $S \mid \Delta$ does not exceed the MEV extractable from S by more than a factor of ε . This notion does not capture our intuition of assessing the security of Δ in terms of the economic losses that Δ could incur due to adversaries interacting with the context S . For example, an airdrop contract Δ that gives away tokens would be deemed insecure, while in reality its interactions with S are irrelevant.

In a different security setting, a similar intuition was the basis of Goguen and Meseguer’s non-interference [11], which was originally formulated as follows:

“One group of users, using a certain set of commands, is noninterfering with another group of users if what the first group does with those commands has no effect on what the second group of users can see”.

In the setting of smart contract compositions, this notion can be reinterpreted by requiring that adversaries interacting with S do not inflict economic damage to Δ . The notion of *MEV non-interference* introduced by [6] is based on this idea, using MEV as a measure of economic damage. The approaches in [14,24] are also based on the idea of non-interference, but replacing MEV with an explicit tagging of contract variables as high-level or low-level variables.

A common aspect of these approaches to economic non-interference is their *qualitative* nature: namely, these definitions classify a composition as either secure or insecure, in a binary fashion. While a qualitative evaluation is sufficient when a composition is deemed secure, in that case that it is not, it does not provide any meaningful estimate of the *degree* of interference. For example, in the insecure composition between a lending protocol and a DEX mentioned above, a quantitative measure could provide insights into the extent to which the system state (e.g., the liquidity reserves in the DEX) and the contract parameters (e.g., the collateralization threshold) contribute to increasing the economic loss.

Contributions This paper introduces a quantitative notion of economic security for smart contract compositions. Our *MEV interference*, which we denote by $\mathcal{J}(S \rightsquigarrow \Delta)$, measures the increase of economic loss of contracts Δ that adversaries can achieve by manipulating the context S . We apply our notion to assess the security of some notable contract compositions, including a bet on a token price, and a lending protocol relying on a DEX as a price oracle. We prove some fundamental properties of our notion: more specifically, $\mathcal{J}(S \rightsquigarrow \Delta)$ increases when S is extended with contracts that are not in the dependencies of Δ (Theorem 1); $\mathcal{J}(S \rightsquigarrow \Delta)$ does not depend on the token balances of users except adversaries (Theorem 2); $\mathcal{J}(S \rightsquigarrow \Delta)$ is preserved when extending S with contracts Γ that enjoy some specific independency conditions with respect to Δ (Theorem 3).

Table 1: Summary of notation.

\mathbf{A}, \mathbf{B}	User accounts	\mathcal{A}, \mathcal{B}	Sets of [user contract] accounts
\mathbf{C}, \mathbf{D}	Contract accounts	\mathcal{C}, \mathcal{D}	Sets of contract accounts
\mathbf{T}, \mathbf{T}'	Token types	$\$1_{\mathbf{T}}$	Price of \mathbf{T}
\mathbf{X}, \mathbf{X}'	Transaction names	$\mathbf{A}:\mathbf{C}.\mathbf{f}(\mathbf{args})$	Transaction
\mathbf{S}, \mathbf{S}'	Blockchain states	$\$_{\mathcal{C}}(\mathbf{S})$	Wealth of contracts \mathcal{C} in \mathbf{S}
\mathbf{W}, \mathbf{W}'	Wallet states	$\mathit{deps}(\mathcal{C})$	Dependencies of contracts \mathcal{C}
$\mathbf{\Gamma}, \mathbf{\Delta}$	Contract states	$\dagger\mathbf{\Gamma}$	Contract accounts in $\mathbf{\Gamma}$

2 Smart contracts model

We consider a contract model inspired by account-based platforms such as Ethereum. The basic building blocks of our model are a set \mathbf{T} of *token types* ($\mathbf{T}, \mathbf{T}', \dots$), representing crypto-assets (e.g., ETH), and a set \mathbf{A} of *accounts*. We partition accounts into *user accounts* $\mathbf{A}, \mathbf{B}, \dots \in \mathbf{A}_u$ (representing the so-called *externally owned accounts* in Ethereum) and *contract accounts* $\mathbf{C}, \mathbf{D}, \dots \in \mathbf{A}_c$.

The state of a user account is a map $w \in \mathbf{T} \rightarrow \mathbb{N}$ from token types to non-negative integers, representing a *wallet* of tokens. The state of a contract account is a pair (w, σ) , where w is a wallet and σ is a key-value map, representing the contract storage. A *blockchain state* \mathbf{S} is a map from accounts to their states. We write an account state in square brackets, wherein we denote by $n:\mathbf{T}$ a balance of n units of token \mathbf{T} in the wallet, and by $\mathbf{x} = v$, the association of value v to the storage variable \mathbf{x} . For example, $\mathbf{C}[1:\mathbf{T}, \mathbf{owner} = \mathbf{A}]$ represents a state where the contract \mathbf{C} stores 1 unit of \mathbf{T} , and the variable \mathbf{owner} contains the address \mathbf{A} . We write a blockchain state as the composition of its account states, using the symbol $|$ as a separator. For example, $\mathbf{S} = \mathbf{A}[1:\mathbf{T}, 2:\mathbf{ETH}] | \mathbf{C}[1:\mathbf{T}, \mathbf{owner} = \mathbf{A}]$ is a state composed by a user account and a contract account.

Contracts are made up of a finite set of *functions*, which can be called by *transactions* sent by users. A function can: (i) receive parameters and tokens from the caller, (ii) transfer tokens to user accounts (including the caller), (iii) update the contract state, (iv) call other functions (possibly of other contracts, and possibly transferring tokens along with the call), (v) return values to the caller. Functions can only manipulate tokens as described above: in particular, they cannot mint or burn tokens, or drain tokens from other accounts. Transactions $\mathbf{X}, \mathbf{X}', \dots$ are calls to contract functions, written $\mathbf{A}:\mathbf{C}.\mathbf{f}(\mathbf{args})$, where \mathbf{A} is the user signing the transaction, \mathbf{C} is the called contract, \mathbf{f} is the called function, and \mathbf{args} is the list of actual parameters. Parameters can also include transfers of tokens \mathbf{T} from \mathbf{A} to \mathbf{C} , written $\mathbf{A} \mathit{pays} n:\mathbf{T}$. Invalid transactions are reverted (i.e., they do not update the blockchain state). We remark that our security definition and results do not rely on a particular language for functions: we just assume a deterministic transition relation \rightarrow between blockchain states, where state transitions are triggered by transactions. To write examples, however, we will instantiate this abstract model using a contract language inspired by Solidity.

We assume that a contract \mathbf{D} can call a function of a contract \mathbf{C} only if \mathbf{C} was deployed before \mathbf{D} . Formally, defining $\mathbf{C} \prec \mathbf{D}$ (read: “ \mathbf{C} is called by \mathbf{D} ”) when some

function in \mathbf{D} calls some function in \mathbf{C} , we require that the transitive and reflexive closure \sqsubseteq of $<$ is a partial order. We define the *dependencies* of a contract \mathbf{C} as $\text{deps}(\mathbf{C}) = \{\mathbf{C}' \mid \mathbf{C}' \sqsubseteq \mathbf{C}\}$, and extend this notion to *sets* of contracts \mathcal{C} . We assume that blockchain states S enjoy the following conditions: (i) S contains all its dependencies, i.e. if \mathbf{C} is a contract in S , then also the contracts $\text{deps}(\mathbf{C})$ are in S ; (ii) S contains *finite tokens*. All states mentioned in our results are assumed to enjoy these well-formedness assumption.³ We write $S = W \mid \Gamma$ for a blockchain state S composed of user wallets W and contract states Γ . We can deconstruct wallets, writing $S = W \mid W' \mid \Gamma$ when the accounts in W and W' are disjoint, as well as contract states, writing $S = W \mid \Gamma \mid \Delta$. We denote by $\dagger\Gamma$ the set of contract accounts in Γ , i.e. $\dagger\Gamma = \text{dom } \Gamma$. For example, $\dagger(\mathbf{C}[\dots] \mid \mathbf{D}[\dots]) = \{\mathbf{C}, \mathbf{D}\}$. Given $\mathbf{X} = \mathbf{A}:\mathbf{C}.\mathbf{f}(\text{args})$, we write $\text{callee}(\mathbf{X})$ for the target contract \mathbf{C} .

3 Threat model

To define economic security of smart contract compositions, following [3] we consider the Maximal Extractable Value (MEV) that can be extracted when new contracts \mathcal{C} are deployed in a blockchain state $S = W \mid \Gamma$, leading to a new state $S \mid \Gamma \mid \Delta$ where Δ contains the initial state of the new contracts \mathcal{C} . Since our goal is measuring the loss of the new contracts Δ caused by attacking their dependencies Γ , rather than considering the overall MEV of $S \mid \Delta$, we isolate the MEV extractable from Δ and compare it to the MEV that could be extracted from Δ *without* exploiting the dependencies Γ . To this purpose, we leverage the adversary model and the notion of *local MEV* introduced in [6].

We start by designating a finite subset \mathcal{M} of user accounts as adversaries. We assume that adversaries have full control of the selection and ordering of transactions — a standard assumption in definitions of MEV [3]. Then, to measure the economic loss of a set of contracts \mathcal{C} , we consider the wealth of \mathcal{C} in a blockchain state before and after the attack. The wealth of \mathcal{C} in S , written $\$_{\mathcal{C}}(S)$, is given by the amount of tokens in each contract $\mathbf{C} \in \mathcal{C}$ in S weighted by their prices. Recalling that a contract state is a pair (w, σ) whose first element is a wallet, and denoting by $\$1_{\mathbf{T}}$ the price of a token type \mathbf{T} , the wealth of a single contract state $\mathbf{C}[w, \sigma]$ is given by $\sum_{\mathbf{T}} w(\mathbf{T}) \cdot \$1_{\mathbf{T}}$, i.e. the summation, for all token types \mathbf{T} , of the number of tokens \mathbf{T} in the wallet of \mathbf{C} , times the price of \mathbf{T} .⁴ By extending this to the set \mathcal{C} , we obtain the following general definition of wealth:

$$\$_{\mathcal{C}}(S) = \sum_{\mathbf{C} \in \mathcal{C}, \mathbf{T}} \text{fst}(\Gamma(\mathbf{C}))(\mathbf{T}) \cdot \$1_{\mathbf{T}} \quad (1)$$

Building on the definition of wealth, we now revisit the notion of local MEV introduced in [6]. The local MEV extractable by a set of contracts \mathcal{C} in a

³ Note that well-formedness rules out some problematic features like reentrancy, which instead is present in Ethereum. However, reentrancy can always be removed by using suitable programming patterns, so we do not consider this as a limitation.

⁴ Here we implicitly assume that the prices of *native* crypto-assets are constant, since they do not depend on the blockchain state. We discuss this assumption in Section 6.

blockchain state S , denoted by $\text{MEV}(S, \mathcal{C})$, is the maximum loss that adversaries can inflict to \mathcal{C} by performing an arbitrary sequence of transactions crafted using their knowledge. By denoting with $\kappa(\mathcal{M})$ the set of transactions craftable by \mathcal{M} , this amounts to the maximum loss $\$_{\mathcal{C}}(S) - \$_{\mathcal{C}}(S')$ over all possible states S' reachable through a sequence \vec{X} of transactions in $\kappa(\mathcal{M})$. In symbols:

$$\text{MEV}(S, \mathcal{C}) = \max \left\{ \$_{\mathcal{C}}(S) - \$_{\mathcal{C}}(S') \mid \vec{X} \in \kappa(\mathcal{M})^*, S \xrightarrow{\vec{X}} S' \right\} \quad (2)$$

In $\text{MEV}(S, \mathcal{C})$, adversaries are allowed to call any contract in S , including the dependencies of \mathcal{C} not defined in \mathcal{C} itself. This follows from the fact that $\kappa(\mathcal{M})$ does not pose any restriction on the callee of the transactions craftable by \mathcal{M} . To estimate the MEV extractable from Δ *without* exploiting the dependencies Γ , we introduce an additional parameter \mathcal{D} to local MEV, representing the set of contracts callable by \mathcal{M} . We denote by $\kappa_{\mathcal{D}}(\mathcal{M}) = \{X \in \kappa(\mathcal{M}) \mid \text{callee}(X) \in \mathcal{D}\}$ the set of transactions craftable by \mathcal{M} and targeting contracts in \mathcal{D} . We define:

$$\text{MEV}_{\mathcal{D}}(S, \mathcal{C}) = \max \left\{ \$_{\mathcal{C}}(S) - \$_{\mathcal{C}}(S') \mid \vec{X} \in \kappa_{\mathcal{D}}(\mathcal{M})^*, S \xrightarrow{\vec{X}} S' \right\} \quad (3)$$

Note that by the finite token assumption in Section 2, the wealth is always finite, and so also the local MEV.

4 A quantitative notion of economic security

In this section we introduce our notion of quantitative security for smart contract compositions, and study its theoretical properties. In Section 5 we will apply it to analyse some archetypal compositions and attacks.

Let S be a blockchain state, formed by users' wallets W and contract states Γ , where we want to deploy new contracts with an initial state Δ . Note that, by the well-formedness assumption introduced in Section 2, the dependencies of Δ must be included in $\Gamma \mid \Delta$, i.e. any function call made by a contract in Δ must target some contracts in Γ or in Δ . We want to measure the security of the composition $S \mid \Delta$ by analysing the additional loss that an adversary can inflict to the contracts in Δ by manipulating the dependencies Γ . To this purpose, our definition will compare:

- $\text{MEV}(S \mid \Delta, \dagger\Delta)$, the maximal loss of the contracts in Δ , where adversaries are able to send transactions to *any* contract in $S \mid \Delta$;
- $\text{MEV}_{\dagger\Delta}(S \mid \Delta, \dagger\Delta)$, the maximal loss of the contracts in Δ , where adversaries can *only* send transactions to contracts in Δ . Note that interactions between Δ and Γ are still possible, as contracts in Δ can invoke functions of contracts in Γ (“*contract dependencies*”), and adversaries can extract tokens from Γ to play them in calls to contracts in Δ (“*token dependencies*”).

Our security notion, called *MEV interference*, measures how leveraging the dependencies in S can amplify the loss caused to Δ . We denote with $\mathcal{J}(S \rightsquigarrow \Delta)$ the MEV interference caused by a blockchain state S to Δ .

Listing 1.1: A simple airdrop contract.

```

contract Airdrop {
  fund/pay x:T { } // any user can deposit x:T to the contract
  withdraw(x) { // any user can withdraw any amount x:T
    require(balance(T)>=x); // check that the contract has at least x:T
    transfer(sender,x:T); // transfer x:T to the caller
  }
}

```

Listing 1.2: A simple airdrop contract with fees.

```

contract AirdropFee {
  fund/pay x:T { } // any user can deposit x:T to the contract
  withdraw(x) { // any user can withdraw any amount x:T (minus fee)
    require(balance(T)>=x);
    fee = floor((FeeManager.getFee() * x) / 100); // integer division
    transfer(sender, x-fee:T);
    transfer(FeeManager.getOwner(), fee:T);
  }
}
contract FeeManager {
  constructor() { owner=sender; feeRate=1; }
  getOwner() { return owner; }
  getFee() { return feeRate; }
  setFee(r) { require(r>=0 && r<=100); feeRate=r; }
}

```

Definition 1 (MEV interference). For a blockchain state S and a contract state Δ , we quantify the MEV interference caused by S on Δ as:

$$\mathcal{J}(S \rightsquigarrow \Delta) = \begin{cases} 1 - \frac{\text{MEV}_{\dagger\Delta}(S \mid \Delta, \dagger\Delta)}{\text{MEV}(S \mid \Delta, \dagger\Delta)} & \text{if } \text{MEV}(S \mid \Delta, \dagger\Delta) \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

Our notion is consistent with the notion of MEV non-interference in [6], which classifies S and Δ as non-interferent if $\text{MEV}_{\dagger\Delta}(S \mid \Delta, \dagger\Delta) = \text{MEV}(S \mid \Delta, \dagger\Delta)$. Namely, $\mathcal{J}(S \rightsquigarrow \Delta) = 0$ iff S and Δ are non-interferent according to [6].

Example 1 (Any/Airdrop). Consider an instance $\Delta = \text{Airdrop}[n:T]$ of the airdrop contract in Listing 1.1, to be deployed in an arbitrary blockchain state S . Note that $\text{MEV}(S \mid \Delta, \{\text{Airdrop}\}) = n \cdot \1_T , since the adversary can craft a transaction $M:\text{Airdrop.withdraw}(n)$ to extract all the tokens from the contract. The restricted $\text{MEV}_{\{\text{Airdrop}\}}(S \mid \Delta, \{\text{Airdrop}\})$ is equal to the unrestricted one, since the adversary just needs to interact with Airdrop . Therefore, if $n > 0$:

$$\mathcal{J}(S \rightsquigarrow \Delta) = 1 - \frac{\text{MEV}_{\{\text{Airdrop}\}}(S \mid \Delta, \{\text{Airdrop}\})}{\text{MEV}(S \mid \Delta, \{\text{Airdrop}\})} = 0$$

The same holds if $n = 0$. This is consistent with our intuition, since the adversary does not need to exploit the dependencies in S to extract MEV from Δ . \diamond

Example 2 (FeeManager/Airdrop). Consider a variant of the airdrop contract, where each withdrawal requires the user to pay a proportional fee (Listing 1.2). To obtain the fee rate, the `AirdropFee` contract calls the `FeeManager` contract. Assume that we want to deploy $\Delta = \text{AirdropFee}[n: \mathbf{T}]$ in a blockchain state S containing `FeeManager[feeRate = r]`. The unrestricted MEV is $n \cdot \$\mathbf{1}_{\mathbf{T}}$, since an adversary can set the fee to 0 by calling `FeeManager.setFee(0)` and then withdraw the full balance of $n: \mathbf{T}$ from `AirdropFee`. Instead, the restricted MEV only amounts to $(n - \lfloor r \cdot n / 100 \rfloor) \cdot \$\mathbf{1}_{\mathbf{T}}$, since the adversary cannot call `FeeManager` to manipulate the fee rate. Therefore, if $n > 0$:

$$\mathcal{J}(S \rightsquigarrow \Delta) = 1 - \frac{n - \lfloor r \cdot n / 100 \rfloor}{n} \leq \frac{r}{100}$$

This is coherent with our intuition: the closer the fee rate is to 100, the greater the difference between restricted and unrestricted MEV, and so the possibility for the attacker to inflict more damage to the contract. \diamond

We now study the theoretical properties of MEV interference. Because of space constraints, we relegate the proofs of our statements to a technical report on ArXiv. Lemma 1 establishes a few basic properties of MEV interference: its value is zero when the context S has no contracts and when Δ is empty; furthermore, the interference is always comprised between 0 and 1.

Lemma 1. (i) $\mathcal{J}(S \rightsquigarrow \emptyset) = 0$; (ii) $\mathcal{J}(W \mid \emptyset \rightsquigarrow \Delta) = 0$; (iii) $0 \leq \mathcal{J}(S \rightsquigarrow \Delta) \leq 1$.

Note that $\mathcal{J}(S \rightsquigarrow \Delta)$ ranges from a minimum 0, representing the case where the context S is not useful to extract MEV from Δ , to a maximum 1, corresponding to the case where the economic loss that can be inflicted to Δ is purely due to the interactions of the adversary with S . Enclosing MEV interference into an interval is a design choice, which we illustrate with an example. Let S be a state with an airdrop contract releasing $1: \mathbf{T}$, where we want to deploy a new contract Δ that, upon the payment of $1: \mathbf{T}$, releases all its balance of $n: \mathbf{ETH}$. Assume that the adversary has no tokens \mathbf{T} , so that she needs to extract $1: \mathbf{T}$ from the airdrop in order to extract MEV from Δ . If we measured the interference from S to Δ as the difference between unrestricted and restricted MEV, i.e.:

$$\mathcal{J}(S \rightsquigarrow \Delta) \stackrel{?}{=} \text{MEV}(S \mid \Delta, \dagger \Delta) - \text{MEV}_{\dagger \Delta}(S \mid \Delta, \dagger \Delta)$$

then we would obtain that $\mathcal{J}(S \rightsquigarrow \Delta) = n \cdot \$\mathbf{1}_{\mathbf{ETH}}$, i.e. the interference would be proportional to the `ETH` balance in Δ . We do not find this measure particularly insightful: after all, what we observe is just that *all* the MEV extractable from Δ is due to the interaction with the context S . In general, under these conditions, our intuition is that the interference should take its maximum value.

Lemma 2 states that when the newly deployed contracts Δ have no wealth (i.e., when $\$_{\dagger \Delta}(\Delta) = 0$), then they have no MEV interference with the context.

Lemma 2. *If $\$_{\dagger \Delta}(\Delta) = 0$, then $\mathcal{J}(S \rightsquigarrow \Delta) = 0$.*

Of course, if Δ has zero wealth, no loss can be inflicted to Δ , regardless of any potential manipulation of its dependencies in S . This also underscores a fundamental aspect of our definition — namely, that it measures what happens in *specific* contract states, rather than in *arbitrary* reachable states of a given contract. For this reason, our intuition is to have $\mathcal{J}(S \rightsquigarrow \Delta) = 0$ whenever Δ has zero wealth, while not ruling out the possibility of having $\mathcal{J}(S' \rightsquigarrow \Delta') > 0$ in a state Δ' where the contracts have been funded.

Theorem 1 says that widening a blockchain state S potentially increases MEV interference to newly deployed contracts Δ . Formally, this amounts to showing that \mathcal{J} is monotonic w.r.t. the operation of adding contracts Γ to the context, i.e. $\mathcal{J}(S \rightsquigarrow \Delta) \leq \mathcal{J}(S \mid \Gamma \rightsquigarrow \Delta)$. Note that by the well-formedness assumption, the statement implicitly assumes that Δ has no dependencies in Γ .

Theorem 1. $\mathcal{J}(S \rightsquigarrow \Delta) \leq \mathcal{J}(S \mid \Gamma \rightsquigarrow \Delta)$

For illustration, consider a state S where we want to deploy new contracts Δ , with an interference estimated as $\mathcal{J}(S \rightsquigarrow \Delta)$. Assume now that the deployment of Δ is front-run by that of another set of contracts Γ . Of course Δ cannot have dependencies in Γ , since otherwise it would not be possible to deploy Δ in S (as this would violate the well-formedness assumption). Now, the interference $\mathcal{J}(S \mid \Gamma \rightsquigarrow \Delta)$ could either be equal to $\mathcal{J}(S \rightsquigarrow \Delta)$, or possibly increase when the adversary can drain tokens from Γ to inflict more loss to Δ . Theorem 1 states that, in any case, the interference should not decrease.

The following example shows a case where the inequality given by Theorem 1 is strict. This is because, even if Δ has no *contract* dependencies in Γ , the adversary may exploit their *token* dependencies, i.e. extract tokens from Γ and leverage them to extract more tokens from Δ .

Example 3. Let $S = \mathbf{M}[0:\mathbf{T}]$ be a state where the adversary has no tokens, and there are no contracts. Consider a contract **Doubler** with a function that, upon receiving as input $n:\mathbf{T}$, returns to the sender $2n:\mathbf{T}$, and let $\Delta = \mathbf{Doubler}[2:\mathbf{T}]$. By Lemma 1, S does not interfere with Δ . Instead, adding $\Gamma = \mathbf{Airdrop}[1:\mathbf{T}]$ to S yields $\mathcal{J}(S \mid \Gamma \rightsquigarrow \Delta) = 1$, since $\text{MEV}_{\{\mathbf{Doubler}\}}(S \mid \Gamma \mid \Delta, \{\mathbf{Doubler}\}) = 0$ while $\text{MEV}(S \mid \Gamma \mid \Delta, \{\mathbf{Doubler}\}) = 2 \cdot \$1_{\mathbf{T}}$. This increase is caused by the ability of \mathbf{M} to leverage the token dependencies between the newly deployed **Airdrop** contract to extract more MEV from **Doubler** than previously possible. \diamond

The previous example also shows that wealthier adversaries not always cause greater interference. Indeed, if $S = \mathbf{M}[1:\mathbf{T}]$, then \mathbf{M} does not need to exploit the **Airdrop** to extract MEV from the **Doubler** contract, since she has enough tokens in her wallet. Of course there are also cases where wealthier adversary can cause more MEV interference: we will see this in Example 5, where a sufficiently wealthy \mathbf{M} can win a bet by producing a price fluctuation in an AMM.

Theorem 2 shows that users' wallets are irrelevant to the evaluation of MEV interference. Namely, $\mathcal{J}(S \rightsquigarrow \Delta)$ is preserved when removing from S all the wallets except those of adversaries. Recall that a wallet state W is a map from

accounts to wallets. Then, in a state $S = W \mid \Gamma$, we just need to consider the restriction of W to the domain \mathcal{M} .

Theorem 2. *If $\text{dom } W_{\mathcal{M}} = \mathcal{M}$, then $\mathcal{J}(W_{\mathcal{M}} \mid W \mid \Gamma \rightsquigarrow \Delta) = \mathcal{J}(W_{\mathcal{M}} \mid \Gamma \rightsquigarrow \Delta)$.*

Here the intuition is that the adversary does not have any control of the tokens in users' wallets, and therefore these tokens play no role in the extraction of MEV from Δ . This assumption highlights a simplification in our attacker model, namely that the mempool of users' transactions is not known by the adversary. Formally, this assumption is visible in the definition of MEV in (3), where the set $\kappa_{\mathcal{D}}(\mathcal{M})$ of transactions craftable by the adversary does not take the mempool as a parameter. Were mempool transactions playable by the adversary, then their success would also depend on the users' wallet, and consequently the MEV interference would possibly depend on them. We discuss this in Section 6.

Theorem 3 provides sufficient conditions under which an adversary \mathcal{M} gains no advantage by front-running the newly deployed contracts Δ with malicious contracts $\Gamma_{\mathcal{M}}$. Condition (i) requires the contracts in $\text{deps}(\Delta)$ to be *sender-agnostic*, i.e. their functions are unaware of the identity of the sender, only being able to use it as a recipient of token transfers. Condition (ii) requires that the contracts in $\text{deps}(\Delta)$ are *token independent* with those in the other contracts (not in $\text{deps}(\Delta)$) which could be possibly exploited by \mathcal{M} . Note that since Definition 1 assumes that states are well-formed, Theorem 3 implicitly assumes that contracts in Δ do not call contracts in $\Gamma_{\mathcal{M}}$. Before stating Theorem 3, we formalise sender-agnosticism and token independence.

Definition 2 (Sender-agnosticism). *A contract C is sender-agnostic if, for all states S and for all transitions that involve an (external or internal) call to C , replacing the caller's address a with any other address b results in the same post-transition state, up to the substitution of a with b .*

In practice, the effect of calling a function of a sender-agnostic contract C can be decomposed into: (i) updating the states of contracts (either directly or through internal calls); (ii) transferring tokens between users and contracts; (iii) transferring tokens to the **sender** of the call to C . Any call to C with the same arguments and **origin**, but distinct **sender**, has exactly the same effect, except for item (iii), where tokens are transferred to the new sender.

Token independence relies on two auxiliary notions: the token types that can be received by contracts Γ from other contracts in S , denoted by $\text{in}_S(\Gamma)$, and those that can be sent from Γ to other contracts, written $\text{out}_S(\Gamma)$.

Definition 3 (Token independence). *Let $S = W \mid \Gamma$, and let $\Delta \preceq \Gamma$ be a subset of the contract states in Γ . We define:*

- $\text{in}_S(\Delta)$ as the set of token types \mathbb{T} for which there exists a state S' reachable from S through a sequence of steps, containing a transaction that causes an inflow of tokens \mathbb{T} from outside Δ to one of the contracts in Δ .

Listing 1.3: An exchange contract.

```

contract Exchange {
  constructor(pay x:tout_, tin_, rate_) { // receive x:tout_ from sender
    require rate_>0 && tin_!=tout_;
    rate=rate_; tout=tout_; tin=tin_; owner=sender;
  }
  getTokens() { return (tin,tout); }
  getRate(tout) { return rate; } // 1:tin for getRate(tout):tout
  setRate(r) { require sender==owner; rate=r; } // sender can update rate

  swap(pay x:tin) { // sender sells x units of token tin
    y = x*getRate(tout); // units of token tout sold to sender
    require balance(tout)>=y; // Exchange has enough tout tokens
    transfer(sender, y:tout); // send y units of token tout to sender
  }
}

```

- $out_S(\Delta)$ as the set of token types \mathbb{T} for which there exists a state S' reachable from S through a sequence of steps, containing a transaction that causes an outflow of tokens \mathbb{T} from Δ to one of the contracts outside Δ .

Let now $\Delta_0 \preceq \Gamma$ and $\Delta_1 \preceq \Gamma$. We say that Δ_0 and Δ_1 are token independent in $S = W \mid \Gamma$ when $in_S(\Delta_0) \cap out_S(\Delta_1) = \emptyset = in_S(\Delta_1) \cap out_S(\Delta_0)$.

Theorem 3. $\mathcal{J}(S \rightsquigarrow \Delta) = \mathcal{J}(S \mid \Gamma_{\mathcal{M}} \rightsquigarrow \Delta)$ holds if (i) the contracts in $deps(\Delta)$ are sender-agnostic, and (ii) $deps(\Delta)$ and $deps(S \mid \Gamma_{\mathcal{M}}) \setminus deps(\Delta)$ are token independent in $S \mid \Gamma_{\mathcal{M}} \mid \Delta$.

Note that $\mathcal{J}(S \rightsquigarrow \Delta)$ is zero when the contract dependencies and the token dependencies of Δ in S are irrelevant to the ability of inflicting a loss to Δ . E.g., consider an arbitrary state S where we want to deploy an airdrop contract Δ (see Listing 1.1). In this scenario, the adversary cannot gain any advantage from the contracts in S , since she can extract the full MEV from the airdrop by interacting with Δ , only. Therefore, the MEV interference from S to Δ is zero.

5 Use cases

We now illustrate MEV interference through a set of use cases. For simplicity, we assume the values in these use cases are real numbers and that all computations are performed using exact arithmetic. We note that adapting our results to smart contract platforms, that, like Ethereum, operate on integers, requires several modifications, such as applying flooring to arithmetic operations and replacing equalities with inequalities. We refer to Appendix B for details.

Example 4 (Airdrop/Exchange). Consider an instance of the `Exchange` contract in Listing 1.3, to be deployed in a blockchain state S containing an instance of the `Airdrop` contract in Listing 1.1. More specifically, let:

$$\begin{aligned}
 S &= \mathbf{M}[n_{\mathcal{M}}:\mathbb{T}] \mid \mathbf{Airdrop}[n_{\mathcal{A}}:\mathbb{T}] \\
 \Delta &= \mathbf{Exchange}[n_{\mathcal{E}}:\mathbf{ETH}, \text{tin} = \mathbb{T}, \text{tout} = \mathbf{ETH}, \text{rate} = r, \text{owner} = \mathbf{A}]
 \end{aligned}$$

The `Exchange` contract allows any user to swap tokens of type `tin` with tokens of type `tout` (in the instance, `T` and `ETH`, respectively), at an exchange rate of 1 unit of `tin` for `rate` units of `tout`. For simplicity, assume that $\$1_T = \$1_{ETH} = 1$. We evaluate the MEV interference from S to Δ . When the exchange rate is favourable, i.e. $r > 1$, the adversary `M` can extract MEV from Δ by exchanging `T` for `ETH`. This is possible as far as `Exchange` has enough `ETH` balance. The MEV can be further increased by draining n_A `T` from `Airdrop`, and swapping these tokens through the `Exchange`. More precisely, we have:

$$\begin{aligned} \text{MEV}_{\{\text{Exchange}\}}(S \mid \Delta, \{\text{Exchange}\}) &= \begin{cases} n_M \cdot r & \text{if } n_M < n_E/r \\ n_E & \text{otherwise} \end{cases} \\ \text{MEV}(S \mid \Delta, \{\text{Exchange}\}) &= \begin{cases} (n_M + n_A) \cdot r & \text{if } n_M < n_E/r - n_A \\ n_E & \text{otherwise} \end{cases} \end{aligned}$$

Therefore, the MEV interference from S on Δ is given by:

$$\mathcal{J}(S \rightsquigarrow \Delta) = \begin{cases} n_A/(n_M+n_A) & \text{if } n_M < n_E/r - n_A \\ 1 - n_M \cdot r/n_E & \text{if } n_E/r - n_A \leq n_M < n_E/r \\ 0 & \text{otherwise} \end{cases}$$

When `M` is sufficiently rich, she can drain the `Exchange` without invoking the `Airdrop`. Instead, when `M`'s wealth is limited, she is able to inflict a greater loss of `Exchange` by leveraging the `Airdrop`. So, the interference caused to `Exchange` in this case has a dual dependence on the adversary's and the `Airdrop`'s wealth. Furthermore, the interference is inversely proportional to `M`'s wealth, i.e. richer adversaries have less need to exploit the context, resulting in lower interference from S to Δ . This is coherent with our intuition, since we would expect a poorer adversary to benefit more from exploiting the `Airdrop` than a richer one. \diamond

Example 5 (AMM/Bet). The `Bet` contract in Listing 1.4 allows a player to bet on the exchange rate between a token and `ETH`. It is parameterized over an `oracle` that is queried for the token price. To enter the bet, the player must match the initial pot set upon deployment. Before the deadline, the player can win a fraction `potShare` of the pot if the oracle exchange rate exceeds or equals `potShare` times the rate. The remaining fraction is taken by the owner. Consider an instance of `Bet` using the `AMM` in Listing 1.5 as a price oracle:

$$\begin{aligned} S &= \text{M}[m: \text{ETH}] \mid \text{AMM}[r_0: \text{ETH}, r_1: \text{T}] \mid \text{block.num} = d - k \mid \dots \\ \Delta &= \text{Bet}[b: \text{ETH}, \text{owner} = \text{A}, \text{tok} = \text{T}, \text{rate} = r, \text{deadline} = d] \end{aligned}$$

When `M` is allowed to leverage `Bet`'s dependency, she can manipulate the `AMM` to influence the internal exchange rate. If `M` has sufficient funds to enter the bet, she can fire the following sequence of transactions, where, in the `swap` transaction, $x = m - b \geq 0$ is the number of `ETH` units sent to the `AMM` and $y = x r_1 / r_0 + x$ is the number of `T` units received (we omit `M`'s wallet for brevity):

$$\begin{aligned}
S \mid \Delta &\xrightarrow{\text{M:Bet.bet(M pays } b:\text{ETH}, p)} \text{AMM}[r_0:\text{ETH}, r_1:\text{T}] \mid \text{Bet}[2b:\text{ETH}, \text{potShare} = p, \dots] \mid \dots \\
&\xrightarrow{\text{M:AMM.swap(M pays } x:\text{ETH}, 0)} \text{AMM}[r_0 + x:\text{ETH}, r_1 - y:\text{T}] \mid \text{Bet}[2b:\text{ETH}, \dots] \mid \dots \\
&\xrightarrow{\text{M:Bet.win()}} \text{AMM}[r_0 + x:\text{ETH}, r_1 - y:\text{T}] \mid \text{Bet}[2b - 2bp:\text{ETH}, \dots] \mid \dots \\
&\xrightarrow{\text{M:AMM.swap(M pays } y:\text{T}, 0)} \text{AMM}[r_0:\text{ETH}, r_1:\text{T}] \mid \text{Bet}[2b - 2bp:\text{ETH}, \dots] \mid \dots
\end{aligned}$$

The bet value that maximizes the loss caused to **Bet** depends on **M**'s wealth, and is given by $p = r_0 + x/r(r_1 - y)$. Assuming **M** enters the bet only for $p \geq 1/2$ (since a smaller proportion makes the bet irrational for her), by Equation (2) we have:

$$\text{MEV}(S \mid \Delta, \{\text{Bet}\}) = (2(r_0 + m - b)^2 / rr_0r_1 - 1) b$$

If **M** can only interact with **Bet**, she is limited to settle on a lower bet value:

$$\text{MEV}_{\{\text{Bet}\}}(S \mid \Delta, \{\text{Bet}\}) = \begin{cases} 2br_0/rr_1 - b - 1 & \text{if } r_0/rr_1 \geq 1/2 \\ 0 & \text{otherwise} \end{cases}$$

Accordingly, MEV interference is estimated through Definition 1 as follows:

$$\mathcal{J}(S \rightsquigarrow \Delta) = \begin{cases} 1 - \frac{2br_0^2 - rr_0r_1(b+1)}{2b(r_0 + m - b)^2 - brr_0r_1} & \text{if } r_0/rr_1 \geq 1/2 \\ 1 & \text{otherwise} \end{cases}$$

We observe maximum interference when **M** exploits the **Bet** by manipulating the **AMM**, which would be impossible by interacting exclusively with **Bet**. Furthermore, the interference value is proportional to the adversarial wealth, as one would anticipate. By contrast, even if **M** was able to empty a portion of the **Bet** by fair play, she can always increase this loss by manipulating the **AMM** (provided she owns adequate funds). Note that in the composition between **Bet** and **Exchange**, the MEV interference is zero, as the adversary cannot manipulate the exchange rate (unless she is the **Exchange** owner). \diamond

Example 6 (AMM/Lending Pool). The contract **LP** in Listing 1.6 implements a simplified lending protocol, where users can deposit and borrow tokens. Borrowing requires users to have a sufficient *collateralization* [13,5]. This value, defined as the ratio between the value of their deposits and that of their debits, is a measure of the borrowing capacity (full versions of lending protocols include a function that allows liquidators to repay loans of under-collateralized borrowers in exchange for part of their collateral). The contract **LP** is parameterized over an **oracle** that is queried for the token prices. Below we analyze a well-known attack where the underlying **oracle** is an **AMM**, which is manipulated by an adversary to increase her borrowing capacity [12,21,5,20,1].

More specifically, consider the following instance, where $\$1_{\text{ETH}} = 1 = \1_{T} , the **AMM** is balanced, and the adversary **M** has not deposited or borrowed tokens yet:

$$S = \text{M}[n:\text{ETH}] \mid \text{AMM}[r:\text{ETH}, r:\text{T}] \quad \Delta = \text{LP}[a:\text{ETH}, b:\text{T}, \text{Cmin} = C_{\min}, \dots]$$

Listing 1.4: A Bet contract.

```

contract Betoracle {
  constructor(pay x:ETH, tok_, deadline_, rate_) {
    require tok_!=ETH && oracle.getTokens()==(ETH,tok_);
    tok=tok_; deadline=deadline_; rate=rate_; owner=sender;
  }
  bet(pay x:ETH, p_) { // sender gives x:ETH to Bet and chooses potShare
    require player==null && x==balance(ETH) && p_>=0 && p_<=1;
    potShare = p_; player=sender;
  }
  win() { // only callable by player before the deadline
    require block.num<=deadline && sender==player;
    if (oracle.getRate(ETH)>=potShare*rate)
      transfer(player, potShare*balance(ETH):ETH);
  }
  close() { // after the deadline, transfer the ETH balance to the owner
    require block.num>deadline;
    transfer(owner, balance(ETH):ETH);
  }
}

```

Listing 1.5: A constant-product AMM contract.

```

contract AMM {
  constructor(pay x0:T0, pay x1:T1) { require x0>0 && x1>0; }

  getTokens() { return (T0,T1); } // token pair

  getRate(tout) { // 1:tin for getRate(tout):tout
    if (tout==T0) { tin=T1 } else { tin=T0 };
    return balance(tout)/balance(tin);
  }
  swap(pay x:tin, ymin) { // sell x:tin to buy at least ymin:tout
    if (tin==T0) { tout=T1 } else { tout=T0 };
    y = x*getRate(tout); // units of token tout sold to sender
    require ymin<=y<balance(tout); // the AMM has enough tout tokens
    transfer(sender, y:tout); // send y units of token tout to sender
  }
}

```

If **M** can interact with the **AMM**, she has the following attack strategy: deposit $(n - x)$:**ETH** to the **LP**, and use the remaining x :**ETH** to inflate the price of **T** in the **AMM**. This allows **M** to increase the amount of **T** she can borrow, since the **LP** now uses an artificially inflated price to determine her borrowing capacity.

To implement this strategy, **M** fires the following sequence of transactions, where we denote by y the amount of **T** units that **M** receives from the **swap**, and with t the amount of **T** units that **M** manages to borrow from the **LP** (below, we omit **M**'s wallet, and the parts of the state that do not change upon a transition):

$$\begin{aligned}
S \mid \Delta & \xrightarrow{M:LP.deposit(M \text{ pays } (n-x):ETH)} AMM[r:ETH, r:T] \mid LP[a + n - x:ETH, b:T, \dots] \mid \dots \\
& \xrightarrow{M:AMM.swap(M \text{ pays } x:ETH, 0)} AMM[r + x:ETH, r - y:T] \mid \dots \\
& \xrightarrow{M:LP.borrow(t,T)} \dots \mid LP[a + n - x:ETH, b - t:T, \dots] \mid \dots \\
& \xrightarrow{M:AMM.swap(M \text{ pays } y:T, 0)} AMM[r:ETH, r:T] \mid \dots
\end{aligned}$$

Listing 1.6: A Lending Pool contract (simplified).

```

contract LPoracle {
  constructor(Cmin_) { Cmin = Cmin_; } // collateralization threshold

  collateral(a) { // return a's collateralization
    val_minted = 0;
    for c in minted: val_minted += minted[t][a] * oracle.getRate(t);
    val_debts = 0;
    for c in debts: val_debts += debt[t][a] * oracle.getRate(t);
    return val_minted / val_debts;
  }

  deposit(a pays x:t) { // a deposits x units of token t in the LP
    minted[t][a] += x; // record the deposited units in the minted map
  }

  borrow(a sig, x, t) { // a borrows x units of token t in the LP
    require balance(t) >= x;
    debts[t][a] += x; // record the borrowed units in the debts map
    require collateral(a) >= Cmin; // a is over-collateralized
    transfer(a, x:t);
  }
}

```

The amount that **M** can borrow (as a function of x) is $t = (n-x)(r+x)^2 / C_{min}(r-y)^2$. Its maximum is obtained for $x = 4n-r/5$ when **M** benefits from the manipulation (i.e., when $4n \geq r$), and for $x = 0$ otherwise.

Assuming that the **LP** has sufficient funds, the unrestricted MEV is given by:

$$\begin{aligned}
 \text{MEV}(S \mid \Delta, \{\text{LP}\}) &= \frac{(n-x)(r+x)^2}{C_{min}(r-y)^2} + x - n \\
 &= \begin{cases} \left(\frac{n+r}{5}\right) \left(\frac{1}{C_{min}} \left(\frac{4(n+r)}{5r}\right)^4 - 1\right) & \text{if } 4n \geq r \\ n \left(\frac{1}{C_{min}} - 1\right) & \text{otherwise} \end{cases}
 \end{aligned}$$

On the contrary, if **M** was restricted to interact with the **LP** only, she suffers a reduced borrowing allowance. By Equation (3) we have:

$$\text{MEV}_{\{\text{LP}\}}(S \mid \Delta, \{\text{LP}\}) = n \left(\frac{1}{C_{min}} - 1 \right)$$

Accordingly, MEV interference is estimated through Definition 1 as follows:

$$\mathcal{J}(S \rightsquigarrow \Delta) = \begin{cases} 1 - \frac{5^5 r^4 n (1 - C_{min})}{(n+r)(4^4 (n+r)^4 - (5r)^4 C_{min})} & \text{if } 4n \geq r \\ 0 & \text{otherwise} \end{cases}$$

In accordance with our expectations, the interference is indeed proportional to the attack capital n of the adversary. Naturally, adversaries with higher manipulation capital experience an increased borrowing capacity. Moreover, the degree of interference is influenced by the **AMM** reserves since the profitability of the attack rests on the cost of manipulating and de-manipulating the **AMM**. \diamond

6 Conclusions

We have proposed a notion of economic security for smart contract compositions, which quantifies the potential economic loss an adversary can inflict on a contract by targeting its dependencies. Below, we discuss some limitations of our approach and directions for future work.

Limitations To keep our theory manageable, we have made a few simplifying assumptions in our model. A first assumption is that the prices of native crypto-assets are constant. Consequently, the amount of MEV interference is not affected by fluctuations of these prices (while they could depend on the prices provided by DEXes, like in Examples 5 and 6). Handling price updates would require to extend blockchain states with a function mapping tokens to their prices. Another assumption is that the local MEV in Equation (3) does not allow adversaries to exploit their knowledge of pending users' transactions (the public *mempool*). The rationale underlying this choice is that, in our vision, MEV interference should be the basis for a static analysis of smart contracts, where dynamic data such as the mempool transactions are not known. Assuming an over-approximation of users' transactions, we could extend our MEV interference by making the mempool a parameter of local MEV, similarly to what done for the theory of MEV in [7].

Future work While some tools exist for detecting price manipulation attacks in DeFi protocols [23,18,22], and others for estimating MEV opportunities [3,4], there remains a gap in addressing general economic attacks on smart contract compositions. A common analysis technique underlying the detection of price manipulation attacks — also employed by some of the tools mentioned above — is *taint analysis*, which aims at identifying potential data flows from low-level to high-level data. In the DeFi setting, this typically corresponds to flows from to functions that influence token prices to functions that transfer tokens. While this technique could potentially be generalised to analyse *qualitative* MEV non-interference, capturing our notion of *quantitative* interference seems to require more advanced techniques. Some inspiration could be drawn from static analysis techniques for information-theoretic interference [9,19,17,2]. We plan to explore this research line in future work. Our blockchain model represents crypto-assets as token types with primitive transfer operations and built-in linearity guarantees preventing asset creation or destruction. In practice, several blockchains including Ethereum do not have native support for custom tokens, but rather require to implement them as smart contracts exposing standard interfaces. This opens the door for attackers to exploit potential discrepancies between these implementations and the standards, possibly leading to MEV [8]. Applying our MEV interference analysis to such compositions is left as future work.

Acknowledgments Work partially supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan (NRRP) funded by the European Union – NextGenerationEU, and by PRIN 2022 NRRP project DeLiCE (F53D23009130001).

References

1. Arora, S., Li, Y., Feng, Y., Xu, J.: SecPLF: Secure protocols for loanable funds against oracle manipulation attacks. In: ACM Asia Conference on Computer and Communications Security (ASIA CCS). ACM (2024). <https://doi.org/10.1145/3634737.3637681>
2. Assaf, M., Naumann, D.A., Signoles, J., Totel, E., Tronel, F.: Hypercollecting semantics and its application to static analysis of information flow. In: ACM SIGPLAN Symposium on Principles of Programming Languages (POPL). pp. 874–887. ACM (2017). <https://doi.org/10.1145/3009837.3009889>
3. Babel, K., Daian, P., Kelkar, M., Juels, A.: Clockwork finance: Automated analysis of economic security in smart contracts. In: IEEE Symposium on Security and Privacy. pp. 622–639. IEEE Computer Society (2023). <https://doi.org/10.1109/SP46215.2023.00036>
4. Babel, K., Javaheripi, M., Ji, Y., Kelkar, M., Koushanfar, F., Juels, A.: Lanturn: Measuring economic security of smart contracts through adaptive learning. In: ACM SIGSAC Conference on Computer and Communications Security (CCS). pp. 1212–1226. ACM (2023). <https://doi.org/10.1145/3576915.3623204>
5. Bartoletti, M., Chiang, J.H., Lluch-Lafuente, A.: SoK: Lending Pools in Decentralized Finance. In: Workshop on Trusted Smart Contracts. LNCS, vol. 12676, pp. 553–578. Springer (2021). https://doi.org/10.1007/978-3-662-63958-0_40
6. Bartoletti, M., Marchesin, R., Zunino, R.: DeFi composability as MEV non-interference. In: Financial Cryptography and Data Security (FC 2024). LNCS, vol. 14745. Springer (2025), https://doi.org/10.1007/978-3-031-78679-2_20
7. Bartoletti, M., Zunino, R.: A theoretical basis for MEV. In: Financial Cryptography and Data Security. LNCS, Springer (2025), to appear
8. Chen, T., Zhang, Y., Li, Z., Luo, X., Wang, T., Cao, R., Xiao, X., Zhang, X.: Tokenscope: Automatically detecting inconsistent behaviors of cryptocurrency tokens in Ethereum. In: ACM SIGSAC Conference on Computer and Communications Security (CCS). pp. 1503–1520. ACM (2019). <https://doi.org/10.1145/3319535.3345664>
9. Clark, D., Hunt, S., Malacaria, P.: A static analysis for quantifying information flow in a simple imperative language. *J. Comput. Secur.* **15**(3), 321–371 (2007). <https://doi.org/10.3233/JCS-2007-15302>
10. 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 Symp. on Security and Privacy. pp. 910–927. IEEE (2020). <https://doi.org/10.1109/SP40000.2020.00040>
11. Goguen, J.A., Meseguer, J.: Security policies and security models. In: IEEE Symposium on Security and Privacy. pp. 11–20. IEEE Computer Society (1982). <https://doi.org/10.1109/SP.1982.10014>
12. Gudgeon, L., Pérez, D., Harz, D., Livshits, B., Gervais, A.: The decentralized financial crisis. In: Crypto Valley Conference on Blockchain Technology (CVCBT). pp. 1–15. IEEE (2020). <https://doi.org/10.1109/CVCBT50464.2020.00005>
13. Gudgeon, L., Werner, S., Perez, D., Knottenbelt, W.J.: DeFi protocols for loanable funds: Interest rates, liquidity and market efficiency. In: ACM Conference on Advances in Financial Technologies (AFT). pp. 92–112 (2020). <https://doi.org/10.1145/3419614.3423254>
14. Guesmi, S., Piazza, C., Rossi, S.: Noninterference analysis for smart contracts: Would you bet on it? In: Distributed Ledger Technology Workshop (DLT). CEUR Workshop Proceedings, vol. 3791. CEUR-WS.org (2024)

15. Kitzler, S., Victor, F., Saggese, P., Haslhofer, B.: A systematic investigation of DeFi compositions in Ethereum. In: Financial Cryptography and Data Security Workshops. LNCS, vol. 13412, pp. 272–279. Springer (2022). https://doi.org/10.1007/978-3-031-32415-4_18
16. Kitzler, S., Victor, F., Saggese, P., Haslhofer, B.: Disentangling Decentralized Finance (DeFi) compositions. ACM Trans. Web **17**(2), 10:1–10:26 (2023). <https://doi.org/10.1145/3532857>
17. Klebanov, V.: Precise quantitative information flow analysis - a symbolic approach. Theoretical Computer Science **538**, 124–139 (2014). <https://doi.org/https://doi.org/10.1016/j.tcs.2014.04.022>
18. Kong, Q., Chen, J., Wang, Y., Jiang, Z., Zheng, Z.: DeFiTainter: Detecting price manipulation vulnerabilities in DeFi protocols. In: ACM SIGSOFT International Symposium on Software Testing and Analysis. p. 1144–1156 (2023). <https://doi.org/10.1145/3597926.3598124>
19. Köpf, B., Rybalchenko, A.: Automation of quantitative information-flow analysis. In: International School on Formal Methods for the Design of Computer, Communication, and Software Systems (SFM). LNCS, vol. 7938, pp. 1–28. Springer (2013). https://doi.org/10.1007/978-3-642-38874-3_1
20. Mackinga, T., Nadahalli, T., Wattenhofer, R.: TWAP oracle attacks: Easier done than said? In: IEEE International Conference on Blockchain and Cryptocurrency (ICBC). pp. 1–8. IEEE (2022). <https://doi.org/10.1109/ICBC54727.2022.9805499>
21. Qin, K., Zhou, L., Livshits, B., Gervais, A.: Attacking the DeFi ecosystem with Flash Loans for fun and profit. In: Financial Cryptography. LNCS, vol. 12674, pp. 3–32. Springer (2021). https://doi.org/10.1007/978-3-662-64322-8_1
22. Wu, K.W.: Strengthening DeFi security: A static analysis approach to Flash Loan vulnerabilities. CoRR **abs/2411.01230** (2025). <https://doi.org/10.48550/arXiv.2411.01230>
23. Wu, S., Wang, D., He, J., Zhou, Y., Wu, L., Yuan, X., He, Q., Ren, K.: DeFiRanger: Detecting price manipulation attacks on defi applications. CoRR **abs/2104.15068** (2021), <https://arxiv.org/abs/2104.15068>
24. Yao, S., Ni, H., Myers, A.C., Cecchetti, E.: SCIF: A language for compositional smart contract security. CoRR **abs/2407.01204** (2024), <https://arxiv.org/abs/2407.01204>

A Proofs: properties of MEV interference

We start by recalling from [6] a few useful properties of local MEV. We define the relation \preceq between contract states as follows:

$$\Gamma \preceq \Delta \iff \forall \mathbf{c} \in \text{dom } \Gamma. \mathbf{c} \in \text{dom } \Delta \wedge \Gamma(\mathbf{c}) = \Delta(\mathbf{c})$$

Therefore, the condition $\Gamma \preceq \Delta$ in Item 3 of Lemma A.1 means that Δ is a widening of the state Γ with other arbitrary contract states.

Lemma A.1 (Basic properties of MEV [6]). *For all $S, \mathcal{C}, \mathcal{D} \subseteq \mathbb{A}_c$:*

1. $\text{MEV}_{\mathcal{D}}(S, \emptyset) = \text{MEV}_{\emptyset}(S, \mathcal{C}) = 0$, $\text{MEV}_{\mathbb{A}_c}(S, \mathbb{A}_c) \geq \text{MEV}(S)$
2. if $\mathcal{D} \subseteq \mathcal{D}'$, then $\text{MEV}_{\mathcal{D}}(S, \mathcal{C}) \leq \text{MEV}_{\mathcal{D}'}(S, \mathcal{C})$
3. $\text{MEV}_{\mathcal{D}}(W \mid \Gamma, \mathcal{C}) \leq \text{MEV}_{\mathcal{D}}(W \mid \Delta, \mathcal{C})$ if $\Gamma \preceq \Delta$
4. $\text{MEV}_{\mathcal{D}}(W \mid \Gamma, \mathcal{C}) = \text{MEV}_{\mathcal{D}}(W \mid \Gamma, \mathcal{C} \cap \dagger\Gamma) = \text{MEV}_{\mathcal{D} \cap \dagger\Gamma}(W \mid \Gamma, \mathcal{C})$
5. $0 \leq \text{MEV}_{\mathcal{D}}(S, \mathcal{C}) \leq \$_{\mathcal{C}}(S)$

Lemma A.2 states that the only user wallets that need to be taken into account to estimate the MEV are those of the adversary. This is because \mathcal{M} has no way to force other users to spend their tokens in the attack sequence.

Lemma A.2 (MEV and adversaries' wallets [6]). *If $\text{dom } W_{\mathcal{M}} = \mathcal{M}$, then*

$$\text{MEV}_{\mathcal{D}}(W_{\mathcal{M}} \mid W \mid \Gamma, \mathcal{C}) = \text{MEV}_{\mathcal{D}}(W_{\mathcal{M}} \mid \Gamma, \mathcal{C})$$

Proof of Lemma 1

For Item (i), by Item 1 of Lemma A.1 we have that $\text{MEV}(S \mid \emptyset, \dagger\emptyset) = 0$. The thesis follows by Definition 1.

For Item (ii), by Item 4 of Lemma A.1 we have:

$$\text{MEV}(W \mid \emptyset \mid \Delta, \dagger\Delta) = \text{MEV}_{\dagger\Delta}(W \mid \emptyset \mid \Delta, \dagger\Delta)$$

which gives us $\mathcal{J}(W \mid \emptyset \rightsquigarrow \Delta) = 0$, and hence we have our thesis.

For Item (iii), there are two cases. If $\text{MEV}(S \mid \Delta, \dagger\Delta) = 0$, then $\mathcal{J}(S \rightsquigarrow \Delta) = 0$ holds by definition. Otherwise, by Items 2 and 5 of Lemma A.1:

$$\begin{aligned} 0 &\leq \text{MEV}_{\dagger\Delta}(S \mid \Delta, \dagger\Delta) \leq \text{MEV}(S \mid \Delta, \dagger\Delta) \\ \implies 0 &\leq \frac{\text{MEV}_{\dagger\Delta}(S \mid \Delta, \dagger\Delta)}{\text{MEV}(S \mid \Delta, \dagger\Delta)} \leq 1 \\ \implies 0 &\leq 1 - \frac{\text{MEV}_{\dagger\Delta}(S \mid \Delta, \dagger\Delta)}{\text{MEV}(S \mid \Delta, \dagger\Delta)} \leq 1 \end{aligned}$$

which implies $0 \leq \mathcal{J}(S \rightsquigarrow \Delta) \leq 1$, giving us our thesis. \square

Proof of Lemma 2

From Items 2 and 5 of Lemma A.1, we have:

$$0 \leq \text{MEV}_{\dagger\Delta}(S \mid \Delta, \dagger\Delta) \leq \text{MEV}(S \mid \Delta, \dagger\Delta) \leq \$_{\dagger\Delta}(\Delta)$$

By hypothesis, $\$_{\dagger\Delta}(\Delta) = 0$. So, by the inequalities above, $\text{MEV}(S \mid \Delta, \dagger\Delta) = 0$. Definition 1 gives the thesis. \square

Definition A.1 (Gain). *The gain of $\mathcal{C} \subseteq \mathbb{A}_c$ when a transaction sequence \vec{X} is fired in S is given by $\gamma_{\mathcal{C}}(S, \vec{X}) = \$_{\mathcal{C}}(S') - \$_{\mathcal{C}}(S)$ if $S \xrightarrow{\vec{X}} S'$.*

Dually, the loss of $\mathcal{C} \subseteq \mathbb{A}_c$ when a transaction sequence \vec{X} is fired in S is given by $-\gamma_{\mathcal{C}}(S, \vec{X}) = \$_{\mathcal{C}}(S) - \$_{\mathcal{C}}(S')$ if $S \xrightarrow{\vec{X}} S'$.

Lemma A.3 states that widening the contract state Γ preserves the MEV extractable from the target contracts. This is because the contracts allowed to be targeted by the adversary, i.e. \mathcal{D} , are not widened. This refines Item 3 of Lemma A.1, giving an equality under the additional assumption $\mathcal{D} \subseteq \dagger\Gamma$.

Lemma A.3. $\text{MEV}_{\mathcal{D}}(W \mid \Gamma, \mathcal{C}) = \text{MEV}_{\mathcal{D}}(W \mid \Delta, \mathcal{C})$ when $\mathcal{D} \subseteq \dagger\Gamma$ and $\Gamma \preceq \Delta$.

Proof. The inequality \leq follows directly from Item 3 of Lemma A.1. For the inequality \geq , assume that Δ is the composition of the contracts Γ with some other contracts $\bar{\Gamma}$, i.e. $\Gamma \preceq \Delta$, $\bar{\Gamma} \preceq \Delta$, and $\Delta \preceq \Gamma \mid \bar{\Gamma}$. Let $\vec{X} \in \kappa_{\mathcal{D}}(\mathcal{M})^*$ be a valid sequence of transactions that maximizes the loss $-\gamma_{\mathcal{C}}(W \mid \Delta, \vec{X})$. Since \vec{X} consists of transactions targeting contracts in $\mathcal{D} \subseteq \dagger\Gamma$ and since, by the well-formedness assumption, there are no internal calls from Γ to $\bar{\Gamma}$, the contracts in $\bar{\Gamma}$ are not affected by \vec{X} . Hence, executing \vec{X} yields a transition of the form:

$$W \mid \Delta \xrightarrow{\vec{X}} W' \mid \Delta' \quad \text{where } \bar{\Gamma} \preceq \Delta'$$

As noted above, \vec{X} does not include any direct/indirect calls to $\dagger\bar{\Gamma}$, and so \vec{X} is also valid in $W \mid \Gamma$. Therefore, we also have some Γ' such that:

$$W \mid \Gamma \xrightarrow{\vec{X}} W' \mid \Gamma'$$

To prove that the loss is constant, observe that:

$$\begin{aligned} \gamma_{\mathcal{C}}(W \mid \Gamma, \vec{X}) &= \$_{\mathcal{C}}(W' \mid \Gamma') - \$_{\mathcal{C}}(W \mid \Gamma) \\ &= \$_{\mathcal{C}}(\Gamma') - \$_{\mathcal{C}}(\Gamma) \\ &= \$_{\mathcal{C}}(\Delta') - \$_{\mathcal{C}}(\bar{\Gamma}) - \$_{\mathcal{C}}(\Delta) + \$_{\mathcal{C}}(\bar{\Gamma}) \\ &= \$_{\mathcal{C}}(\Delta') - \$_{\mathcal{C}}(\Delta) \\ &= \$_{\mathcal{C}}(W' \mid \Delta') - \$_{\mathcal{C}}(W \mid \Delta) \\ &= \gamma_{\mathcal{C}}(W \mid \Delta, \vec{X}) \end{aligned}$$

This implies that:

$$\text{MEV}_{\mathcal{D}}(W \mid \Delta, \mathcal{C}) \leq \text{MEV}_{\mathcal{D}}(W \mid \Gamma, \mathcal{C})$$

which gives our thesis. \square

Proof of Theorem 1

By Definition 1, we have two cases.

If $\text{MEV}(S \mid \Delta, \dagger\Delta) = 0$, then $\mathcal{J}(S \rightsquigarrow \Delta) = 0$. From Lemma 1(iii), we have $\mathcal{J}(S \mid \Gamma \rightsquigarrow \Delta) \geq 0$. This implies the thesis, $\mathcal{J}(S \rightsquigarrow \Delta) \leq \mathcal{J}(S \mid \Gamma \rightsquigarrow \Delta)$.

Otherwise, assume that $\text{MEV}(S \mid \Delta, \dagger\Delta) > 0$. Then, by Definition 1:

$$\mathcal{J}(S \rightsquigarrow \Delta) = 1 - \frac{\text{MEV}_{\dagger\Delta}(S \mid \Delta, \dagger\Delta)}{\text{MEV}(S \mid \Delta, \dagger\Delta)}$$

Now, by Item 3 of Lemma A.1, we have that:

$$0 < \text{MEV}(S \mid \Delta, \dagger\Delta) \leq \text{MEV}(S \mid \Gamma \mid \Delta, \dagger\Delta)$$

Therefore, by Definition 1:

$$\mathcal{J}(S \mid \Gamma \rightsquigarrow \Delta) = 1 - \frac{\text{MEV}_{\dagger\Delta}(S \mid \Gamma \mid \Delta, \dagger\Delta)}{\text{MEV}(S \mid \Gamma \mid \Delta, \dagger\Delta)}$$

From Lemma A.1, we have that:

$$\begin{aligned} \text{MEV}_{\dagger(S \mid \Delta)}(S \mid \Delta, \dagger\Delta) &\leq \text{MEV}_{\dagger(S \mid \Delta)}(S \mid \Gamma \mid \Delta, \dagger\Delta) && \text{by Item 3} \\ &\leq \text{MEV}_{\dagger(S \mid \Gamma \mid \Delta)}(S \mid \Gamma \mid \Delta, \dagger\Delta) && \text{by Item 2} \end{aligned} \quad (4)$$

We know from (4),

$$\text{MEV}(S \mid \Delta, \dagger\Delta) \leq \text{MEV}(S \mid \Gamma \mid \Delta, \dagger\Delta)$$

Taking the reciprocal on both sides gives us:

$$\frac{1}{\text{MEV}(S \mid \Delta, \dagger\Delta)} \geq \frac{1}{\text{MEV}(S \mid \Gamma \mid \Delta, \dagger\Delta)}$$

By Lemma A.3, we have $\text{MEV}_{\dagger\Delta}(S \mid \Delta, \dagger\Delta) = \text{MEV}_{\dagger\Delta}(S \mid \Gamma \mid \Delta, \dagger\Delta)$. Then:

$$\frac{\text{MEV}_{\dagger\Delta}(S \mid \Delta, \dagger\Delta)}{\text{MEV}(S \mid \Delta, \dagger\Delta)} \geq \frac{\text{MEV}_{\dagger\Delta}(S \mid \Gamma \mid \Delta, \dagger\Delta)}{\text{MEV}(S \mid \Gamma \mid \Delta, \dagger\Delta)}$$

which finally gives us:

$$1 - \frac{\text{MEV}_{\dagger\Delta}(S \mid \Delta, \dagger\Delta)}{\text{MEV}(S \mid \Delta, \dagger\Delta)} \leq 1 - \frac{\text{MEV}_{\dagger\Delta}(S \mid \Gamma \mid \Delta, \dagger\Delta)}{\text{MEV}(S \mid \Gamma \mid \Delta, \dagger\Delta)}$$

which gives our thesis, i.e. $\mathcal{J}(S \rightsquigarrow \Delta) \leq \mathcal{J}(S \mid \Gamma \rightsquigarrow \Delta)$. □

Proof of Theorem 2

By Lemma A.2 we have that, for all \mathcal{C}, \mathcal{D} and for all Γ' :

$$\text{dom } W_{\mathcal{M}} = \mathcal{M} \implies \text{MEV}_{\mathcal{D}}(W_{\mathcal{M}} \mid W \mid \Gamma', \mathcal{C}) = \text{MEV}_{\mathcal{D}}(W_{\mathcal{M}} \mid \Gamma', \mathcal{C})$$

In particular, by choosing $\Gamma' = \Gamma \mid \Delta$ and $\mathcal{C} = \dagger\Delta$, this implies that:

$$\begin{aligned} \text{MEV}(W_{\mathcal{M}} \mid W \mid \Gamma \mid \Delta, \dagger\Delta) &= \text{MEV}(W_{\mathcal{M}} \mid \Gamma \mid \Delta, \dagger\Delta) \\ \text{MEV}_{\dagger\Delta}(W_{\mathcal{M}} \mid W \mid \Gamma \mid \Delta, \dagger\Delta) &= \text{MEV}_{\dagger\Delta}(W_{\mathcal{M}} \mid \Gamma \mid \Delta, \dagger\Delta) \end{aligned}$$

which gives us our thesis, i.e. $\mathcal{J}(W_{\mathcal{M}} \mid W \mid \Gamma \rightsquigarrow \Delta) = \mathcal{J}(W_{\mathcal{M}} \mid \Gamma \rightsquigarrow \Delta)$ \square

Lemma A.4 gives sufficient conditions under which we can strip \mathcal{D} from all the non-dependencies of \mathcal{C} while preserving $\text{MEV}_{\mathcal{D}}(S, \mathcal{C})$. Condition 1 is that contract functions are sender-agnostic, i.e. they are not aware of the identity of the **sender**, being only able to use it as a recipient of token transfers. Condition 2 ensures that \mathcal{D} contains enough contracts to reproduce attacks in the stripped state. Condition 3 requires that the dependencies and the non-dependencies of \mathcal{C} in \mathcal{D} are token independent in S . In other words, there are no token dependencies between $\mathcal{D} \cap \text{deps}(\mathcal{C})$ and $\mathcal{D} \setminus \text{deps}(\mathcal{C})$, which could have potentially be exploited by non-wealthy adversaries.

Lemma A.4. *The equality:*

$$\text{MEV}_{\mathcal{D}}(S, \mathcal{C}) = \text{MEV}_{\mathcal{D} \cap \text{deps}(\mathcal{C})}(S, \mathcal{C})$$

holds if all the following conditions, where $\mathcal{C}' = \text{deps}(\mathcal{C}) \cap \text{deps}(\mathcal{D} \setminus \text{deps}(\mathcal{C}))$, are satisfied:

1. the contracts in \mathcal{C}' are sender-agnostic;
2. $\mathcal{C}' \subseteq \mathcal{D}$;
3. $\text{deps}(\mathcal{D}) \cap \text{deps}(\mathcal{C})$ and $\text{deps}(\mathcal{D}) \setminus \text{deps}(\mathcal{C})$ are token independent in S .

Proof. First, note that the inequality $\text{MEV}_{\mathcal{D} \cap \text{deps}(\mathcal{C})}(S, \mathcal{C}) \leq \text{MEV}_{\mathcal{D}}(S, \mathcal{C})$ follows from Item 2 of Lemma A.1, so we just need to show that:

$$\text{MEV}_{\mathcal{D}}(S, \mathcal{C}) \leq \text{MEV}_{\mathcal{D} \cap \text{deps}(\mathcal{C})}(S, \mathcal{C})$$

To do so, let $\vec{X} \in \kappa_{\mathcal{D}}(\mathcal{M})^*$ be a sequence of transactions that maximizes the loss of \mathcal{C} when executed in state S . We show that there exists $\vec{Y} \in \kappa_{\mathcal{D} \cap \text{deps}(\mathcal{C})}(\mathcal{M})^*$ that causes a loss to \mathcal{C} equal to the one caused by \vec{X} , i.e.:

$$\vec{Y} \in \kappa_{\mathcal{D} \cap \text{deps}(\mathcal{C})}(\mathcal{M})^* \quad \gamma_{\mathcal{C}}(S, \vec{Y}) = \gamma_{\mathcal{C}}(S, \vec{X}) \quad (5)$$

W.l.o.g. we assume that all the transactions in \vec{X} are valid: indeed, invalid transactions in \vec{X} are reverted, so they can be removed without affecting the loss.

Note that each transaction $X_i = M_i: C_{i,1}.f_{i,1}(\text{args}_{i,1})$ in \vec{X} can trigger a sequence of *internal* contract-to-contract function calls:

$$C_{i,1}:C_{i,2}.f_{i,2}(\text{args}_{i,2}) \ C_{i,2}:C_{i,3}.f_{i,3}(\text{args}_{i,3}) \ \cdots \ C_{i,k-1}:C_{i,k}.f_{i,k}(\text{args}_{i,k})$$

Let \vec{x} be the sequence of all function calls (either external or internal) that are performed upon the execution of \vec{X} in state S . To construct \vec{Y} , we start by considering the subsequence \vec{y} of \vec{x} containing all and only the calls of the form:

- (a) $M_i: C_{i,1}.f_{i,1}(\text{args}_{i,1})$ where $C_{i,1} \in \text{deps}(\mathcal{C})$, or
- (b) $C_{i,j-1}: C_{i,j}.f_{i,j}(\text{args}_{i,j})$, where $C_{i,j-1} \notin \text{deps}(\mathcal{C})$ and $C_{i,j} \in \text{deps}(\mathcal{C})$.

Claim (1). If $C_{i,j-1}: C_{i,j}.f_{i,j}(\text{args}_{i,j}) \in \vec{y}$, then $C_{i,j} \in \mathcal{C}'$.

Proof of Claim (1). By hypothesis, $C_{i,j} \in \text{deps}(\mathcal{C})$. Let $X_i \in \vec{X}$ be the transaction that originated the call. Since $X_i \in \kappa_{\mathcal{D}}(\mathcal{M})$, then $C_{i,1} \in \mathcal{D}$. Since $\text{deps}(\mathcal{C})$ is closed downward and $C_{i,j-1} \notin \text{deps}(\mathcal{C})$, then $C_{i,1} \notin \text{deps}(\mathcal{C})$. So, $C_{i,1} \in \mathcal{D} \setminus \text{deps}(\mathcal{C})$, and therefore $C_{i,j} \in \text{deps}(\mathcal{D} \setminus \text{deps}(\mathcal{C}))$. This completes the proof of Claim (1).

To describe the construction of \vec{Y} , let the meta-variables \mathbf{a}_i range over user and contract addresses, so to rewrite the sequence \vec{y} as follows:

$$\mathbf{a}_1: C_1.f_1(\text{args}_1) \ \mathbf{a}_2: C_2.f_2(\text{args}_2) \ \cdots \ \mathbf{a}_n: C_n.f_n(\text{args}_n) \ \cdots$$

We translate \vec{y} into the sequence of *transactions* \vec{Y} by preserving the senders \mathbf{a}_i that are user accounts (i.e., $\mathbf{a}_i = M_i$), and by replacing the \mathbf{a}_i that are contract accounts into the user account that originated the corresponding call. Namely, if $\mathbf{a}_i = C_{i,j-1}$ is a contract account corresponding to the following call in \vec{y} :

$$C_{i,j-1}: C_{i,j}.f_{i,j}(\text{args}_{i,j})$$

then the sender of the i -th transaction in \vec{Y} is M_i , i.e. the originator of the call. Note that each transaction Y_i in \vec{Y} can be funded by the adversary:

- if $\mathbf{a}_i = M_i$, then the fact that the corresponding transaction X_i in \vec{X} was valid implies that M_i has the tokens needed to fund the call;
- if $\mathbf{a}_i = C_{i,j-1}$, then there is no token transfer from $C_{i,j-1}$ to $C_{i,j}$, and so Y_i does not need to be funded. This is because:
 - $C_{i,j-1} \in \text{deps}(\mathcal{D}) \setminus \text{deps}(\mathcal{C})$: indeed, $C_{i,j-1} \in \text{deps}(\mathcal{D})$ since $X_i \in \kappa_{\mathcal{D}}(\mathcal{M})$, and $C_{i,j-1} \notin \text{deps}(\mathcal{C})$ by definition of case (b);
 - $C_{i,j} \in \text{deps}(\mathcal{D}) \cap \text{deps}(\mathcal{C})$: indeed, $C_{i,j} \in \text{deps}(\mathcal{D})$ since $X_i \in \kappa_{\mathcal{D}}(\mathcal{M})$, and $C_{i,j} \in \text{deps}(\mathcal{C})$ by definition of case (b);
 - $\text{deps}(\mathcal{D}) \cap \text{deps}(\mathcal{C})$ and $\text{deps}(\mathcal{D}) \setminus \text{deps}(\mathcal{C})$ are token independent in S by assumption (3).

Claim (2). $\vec{Y} \in \kappa_{\mathcal{D} \cap \text{deps}(\mathcal{C})}(\mathcal{M})^*$

Proof of Claim (2). Consider a transaction Y_i in \vec{Y} . We have two cases, depending on whether Y_i is due to conditions (a) or (b):

- (a) in this case, Y_i corresponds to some $X_i = M_i: C_{i,1}.f_{i,1}(\text{args}_{i,1})$ in \vec{X} where $C_{i,1} \in \text{deps}(\mathcal{C})$. Since $X_i \in \kappa_{\mathcal{D}}(\mathcal{M})$, then $Y_i \in \kappa_{\mathcal{D} \cap \text{deps}(\mathcal{C})}(\mathcal{M})$.
- (b) by Claim (1), the callee of Y_i is in $\mathcal{C}' = \text{deps}(\mathcal{C}) \cap \text{deps}(\mathcal{D} \setminus \text{deps}(\mathcal{C}))$, which is included in \mathcal{D} by assumption 2. Note that \mathcal{M} is able to craft the actual arguments of that call by simulating the execution of \vec{X} . This implies that $Y_i \in \kappa_{\mathcal{D} \cap \text{deps}(\mathcal{C})}(\mathcal{M})$. This completes the proof of Claim (2).

We now show that \vec{Y} and \vec{X} modify the state of contracts in \mathcal{C} in exactly the same way. Note that the transactions Y_i that are in \vec{Y} due to condition (b) have callee in \mathcal{C}' by Claim (1), and so their functions are *sender-agnostic* by assumption 1. So, the fact that in the execution of Y_i they are called directly from a user address, while in the execution of X_i they are called from a contract address, does not affect the execution of these calls. Note that a call $C_{i,j-1}: C_{i,j}.f_{i,j}(\text{args}_{i,j})$ in X_i could send tokens to the sender $C_{i,j-1}$, thus affecting its gain, while the corresponding call $M_i: C_{i,j}.f_{i,j}(\text{args}_{i,j})$ would send these tokens to M_i . This difference however do not affect the gains and losses of \mathcal{C} , since $C_{i,j-1}$ is not in $\text{deps}(\mathcal{C})$ by condition (b).

Note that the sequence \vec{h} of calls performed upon the execution of \vec{Y} contains \vec{y} but does not coincide with it, since it also includes all the internal calls that are performed by functions in \vec{y} . In fact, \vec{h} is the subsequence of \vec{x} that contains every call to functions of contracts in $\text{deps}(\mathcal{C})$. For this reason, both \vec{x} and \vec{h} modify the state of contracts $\text{deps}(\mathcal{C})$ in the same way — and, in particular, they cause exactly the same losses to the contracts in \mathcal{C} . This implies that \vec{Y} is valid in S and that $\gamma_{\mathcal{C}}(S, \vec{Y}) = \gamma_{\mathcal{C}}(S, \vec{X})$. Since we have proved (5) for all possible \vec{X} , we obtain the thesis. \square

Example A.1. To illustrate Lemma A.4, consider the contracts:

```
contract C0 { f(a pays 1:T) { transfer(M, 2:T) } }
contract C1 { f() { C0.f(C1 pays 1:T); } }
contract C2 { f() { require sender==C3; C1.f(); } }
contract C3 { f() { C2.f(); C1.f(); } }
```

Let $\mathcal{M} = \{M\}$, $\mathcal{D} = \{C0, C1, C3\}$, $\mathcal{C} = \{C0, C1\}$, and let:

$$S = M[0:T] \mid C0[2:T] \mid C1[2:T] \mid C2[0:T] \mid C3[0:T]$$

Let $\vec{X} \in \kappa_{\mathcal{D}}(\mathcal{M})$ be the following sequence of transactions:

$$\vec{X} = M:C3.f()$$

By executing \vec{X} in S , we have that:

$$S \xrightarrow{M:C3.f()} M[4:T] \mid C0[0:T] \mid C1[0:T] \mid C2[0:T] \mid C3[0:T]$$

Since there are no tokens left in \mathcal{C} , \vec{X} clearly maximises the loss of \mathcal{C} , hence:

$$\text{MEV}_{\mathcal{D}}(S, \mathcal{C}) = 4 \cdot \$1_T$$

We first check that the conditions of Lemma A.4 are satisfied. Let:

$$\begin{aligned}\mathcal{C}' &= \text{deps}(\mathcal{C}) \cap \text{deps}(\mathcal{D} \setminus \text{deps}(\mathcal{C})) = \{\mathbf{C0}, \mathbf{C1}\} \cap \text{deps}(\{\mathbf{C0}, \mathbf{C1}, \mathbf{C3}\} \setminus \{\mathbf{C0}, \mathbf{C1}\}) \\ &= \{\mathbf{C0}, \mathbf{C1}\} \cap \text{deps}(\{\mathbf{C3}\}) = \{\mathbf{C0}, \mathbf{C1}\} \cap \{\mathbf{C0}, \mathbf{C1}, \mathbf{C2}, \mathbf{C3}\} \\ &= \{\mathbf{C0}, \mathbf{C1}\}\end{aligned}$$

The conditions of Lemma A.4 are then satisfied, since:

- (1) the contracts $\mathbf{C0}, \mathbf{C1} \in \mathcal{C}'$ is sender-agnostic. Note that $\mathbf{C2}$ is not sender-agnostic, but this does not violate assumption 1 since sender-agnosticism is only required on \mathcal{C}' ;
- (2) $\mathcal{C}' = \{\mathbf{C0}, \mathbf{C1}\} \subseteq \mathcal{D}$. Note that this inclusion is stricter than necessary: indeed, in this example, choosing $\mathcal{D} = \{\mathbf{C1}, \mathbf{C3}\}$ would have violated assumption 2, but it would have still preserved the MEV (see \vec{Y} below).
- (3) token independence of the parts of S related to contracts:

$$\begin{aligned}\text{deps}(\mathcal{D}) \cap \text{deps}(\mathcal{C}) &= \{\mathbf{C0}, \mathbf{C1}, \mathbf{C2}, \mathbf{C3}\} \cap \{\mathbf{C0}, \mathbf{C1}\} = \{\mathbf{C0}, \mathbf{C1}\} \\ \text{deps}(\mathcal{D}) \setminus \text{deps}(\mathcal{C}) &= \{\mathbf{C0}, \mathbf{C1}, \mathbf{C2}, \mathbf{C3}\} \setminus \{\mathbf{C0}, \mathbf{C1}\} = \{\mathbf{C2}, \mathbf{C3}\}\end{aligned}$$

Note instead that token independence is not required between $\mathbf{C0}$ and $\mathbf{C1}$: actually, these two contracts are token dependent, since $\mathbf{C1}$ sends $1:\mathbf{T}$ along with the internal call to $\mathbf{C0}$.

We now construct the sequence of transactions \vec{Y} following the proof of Lemma A.4. The sequence \vec{x} of calls induced by \vec{X} , the subsequence \vec{y} obtained by filtering \vec{x} , and the sequence of transactions \vec{Y} are the following:

$$\begin{aligned}\vec{x} &= \text{M: C3.f}() \quad \text{C3: C2.f}() \quad \text{C2: C1.f}() \quad \text{C1: C0.f}() \quad \text{C3: C1.f}() \quad \text{C1: C0.f}() \\ \vec{y} &= \quad \quad \quad \text{C2: C1.f}() \quad \quad \quad \text{C3: C1.f}() \\ \vec{Y} &= \quad \quad \quad \text{M: C1.f}() \quad \quad \quad \text{M: C1.f}()\end{aligned}$$

Note that $\vec{Y} \in \kappa_{\mathcal{D} \cap \text{deps}(\mathcal{C})}(\mathbf{M}) = \kappa_{\{\mathbf{C0}, \mathbf{C1}\}}(\mathbf{M})$. By executing \vec{Y} in S , we have that:

$$\begin{aligned}S &\xrightarrow{\text{M: C1.f}()} \text{M}[2:\mathbf{T}] \mid \text{C0}[1:\mathbf{T}] \mid \text{C1}[1:\mathbf{T}] \mid \text{C2}[0:\mathbf{T}] \mid \text{C3}[0:\mathbf{T}] \\ &\xrightarrow{\text{M: C1.f}()} \text{M}[4:\mathbf{T}] \mid \text{C0}[0:\mathbf{T}] \mid \text{C1}[0:\mathbf{T}] \mid \text{C2}[0:\mathbf{T}] \mid \text{C3}[0:\mathbf{T}]\end{aligned}$$

Hence, we have that:

$$\text{MEV}_{\mathcal{D} \cap \text{deps}(\mathcal{C})}(S, \mathcal{C}) = 4 \cdot \$1_{\mathbf{T}}$$

which confirms the preservation of MEV stated by Lemma A.4. \diamond

Example A.2. To illustrate the need of the token independence assumption in Lemma A.4, consider the contracts:

```
contract C0 { f(a pays 2:T) { transfer(M, 4:T) } }
contract C1 {
```

```

    f() { C0.f(C1 pay 2:T); }
    receive(a pays n:T) { }
}
contract C2 {
    f() { C1.receive(C2 pays 1:T); }
    // transfer(C1,1:T) is forbidden in our model
    // transfer recipients must be user accounts
}
contract C3 { f() { C2.f(); C1.f(); } }

```

Let $\mathcal{M} = \{\mathbf{M}\}$, $\mathcal{C} = \{\mathbf{C0}, \mathbf{C1}\}$, $\mathcal{D} = \{\mathbf{C0}, \mathbf{C1}, \mathbf{C3}\}$, and let:

$$S = \mathbf{M}[0:\mathbf{T}] \mid \mathbf{C0}[2:\mathbf{T}] \mid \mathbf{C1}[1:\mathbf{T}] \mid \mathbf{C2}[1:\mathbf{T}] \mid \mathbf{C3}[0:\mathbf{T}]$$

Note that \mathbf{M} has no tokens in S , so the only way to extract MEV is to pass through $\mathbf{C3}$. Let $\vec{X} \in \kappa_{\mathcal{D}}(\mathbf{M})$ be the following sequence of transactions:

$$\vec{X} = \mathbf{M}:\mathbf{C3}.f()$$

By executing \vec{X} in S , we have that:

$$S \xrightarrow{\mathbf{M}:\mathbf{C3}.f()} \mathbf{M}[4:\mathbf{T}] \mid \mathbf{C0}[0:\mathbf{T}] \mid \mathbf{C1}[0:\mathbf{T}] \mid \mathbf{C2}[0:\mathbf{T}] \mid \mathbf{C3}[0:\mathbf{T}]$$

Since there are no tokens left in \mathcal{C} , \vec{X} clearly maximises the loss of \mathcal{C} . Since \mathcal{C} contained $3:\mathbf{T}$ in S , then:

$$\text{MEV}_{\mathcal{D}}(S, \mathcal{C}) = 3 \cdot \$1_{\mathbf{T}}$$

Similarly to Example A.1, we have that $\mathcal{C}' = \{\mathbf{C0}, \mathbf{C1}\}$, which satisfies conditions (1) and (2). For condition (3) (token independence), we have that:

$$\begin{aligned} \text{deps}(\mathcal{D}) \cap \text{deps}(\mathcal{C}) &= \{\mathbf{C0}, \mathbf{C1}, \mathbf{C2}, \mathbf{C3}\} \cap \{\mathbf{C0}, \mathbf{C1}\} = \{\mathbf{C0}, \mathbf{C1}\} \\ \text{deps}(\mathcal{D}) \setminus \text{deps}(\mathcal{C}) &= \{\mathbf{C0}, \mathbf{C1}, \mathbf{C2}, \mathbf{C3}\} \setminus \{\mathbf{C0}, \mathbf{C1}\} = \{\mathbf{C2}, \mathbf{C3}\} \end{aligned}$$

Now, token independence between $\{\mathbf{C0}, \mathbf{C1}\}$ and $\{\mathbf{C2}, \mathbf{C3}\}$ is not satisfied, since $\mathbf{C2}$ sends $1:\mathbf{T}$ to $\mathbf{C1}$. More formally, we have that:

$$\text{in}_S(\Gamma_{\mathbf{C0}, \mathbf{C1}}) = \{\mathbf{T}\} \quad \text{out}_S(\Gamma_{\mathbf{C2}, \mathbf{C3}}) = \{\mathbf{T}\} \quad \text{in}_S(\Gamma_{\mathbf{C2}, \mathbf{C3}}) = \emptyset \quad \text{out}_S(\Gamma_{\mathbf{C0}, \mathbf{C1}}) = \{\mathbf{T}\}$$

By Definition 3, since:

$$\text{in}_S(\Gamma_{\mathbf{C0}, \mathbf{C1}}) \cap \text{out}_S(\Gamma_{\mathbf{C2}, \mathbf{C3}}) = \{\mathbf{T}\} \cap \{\mathbf{T}\} \neq \emptyset$$

then, $\Gamma_{\mathbf{C0}, \mathbf{C1}}$ and $\Gamma_{\mathbf{C2}, \mathbf{C3}}$ are *not* token independent.

Since the conditions of Lemma A.4 are *not* satisfied, we are not guaranteed to have the preservation of MEV:

$$\text{MEV}_{\mathcal{D} \cap \text{deps}(\mathcal{C})}(S, \mathcal{C}) \stackrel{?}{=} \text{MEV}_{\mathcal{D}}(S, \mathcal{C}) = 3 \cdot \$1_{\mathbf{T}}$$

Indeed, the maximum loss that \mathcal{M} can inflict to \mathcal{C} using $\kappa_{\mathcal{D} \cap \text{deps}(\mathcal{C})}(\mathcal{M}) = \kappa_{\{\mathbf{C0}, \mathbf{C1}\}}(\mathcal{M})$ is zero. This is because:

- calling **C0** fails, since **C0** has not the required 4: **T** to transfer;
- calling **C1** fails, since **C1** does not have the 2: **T** required to call **C0**.

Note also that requiring the milder condition that $\mathcal{D} \cap \text{deps}(\mathcal{C})$ and $\mathcal{D} \setminus \text{deps}(\mathcal{C})$ are token independent would not be enough to guarantee MEV preservation. In our example, we would have:

$$\begin{aligned}\mathcal{D} \cap \text{deps}(\mathcal{C}) &= \{\mathbf{C0}, \mathbf{C1}, \mathbf{C3}\} \cap \{\mathbf{C0}, \mathbf{C1}\} = \{\mathbf{C0}, \mathbf{C1}\} \\ \mathcal{D} \setminus \text{deps}(\mathcal{C}) &= \{\mathbf{C0}, \mathbf{C1}, \mathbf{C3}\} \setminus \{\mathbf{C0}, \mathbf{C1}\} = \{\mathbf{C3}\}\end{aligned}$$

where $\Gamma_{\mathbf{C0}, \mathbf{C1}}$ and $\Gamma_{\mathbf{C3}}$ are token independent. \diamond

Example A.3. To illustrate the need of the sender-agnosticism assumption in Lemma A.4, consider the contracts:

```
contract C0 { f() { require sender==C1; transfer(M, 1:T) } }
contract C1 { f() { C0.f(); } }
```

Let $\mathcal{M} = \{\mathbf{M}\}$, $\mathcal{C} = \{\mathbf{C0}\}$, $\mathcal{D} = \{\mathbf{C0}, \mathbf{C1}\}$, and let:

$$S = \mathbf{M}[0:\mathbf{T}] \mid \mathbf{C0}[1:\mathbf{T}] \mid \mathbf{C1}[0:\mathbf{T}]$$

Let $\vec{X} \in \kappa_{\mathcal{D}}(\mathcal{M})$ be the following sequence of transactions:

$$\vec{X} = \mathbf{M}:\mathbf{C1}.f()$$

By executing \vec{X} in S , we have that:

$$S \xrightarrow{\mathbf{M}:\mathbf{C1}.f()} \mathbf{M}[1:\mathbf{T}] \mid \mathbf{C0}[0:\mathbf{T}] \mid \mathbf{C1}[0:\mathbf{T}]$$

Since there are no tokens left in \mathcal{C} , \vec{X} clearly maximises the loss of \mathcal{C} , hence:

$$\text{MEV}_{\mathcal{D}}(S, \mathcal{C}) = 1 \cdot \$1_{\mathbf{T}}$$

We have that $\mathcal{C}' = \{\mathbf{C0}\}$, which satisfies condition (2). Note that condition (3) (token independence) is trivially satisfied, since there are no token transfers among the contracts. Instead, the contract $\mathbf{C0} \in \mathcal{C}'$ is not sender-agnostic, thus violating condition (1). Indeed, MEV preservation does not hold, since:

$$\text{MEV}_{\mathcal{D} \cap \text{deps}(\mathcal{C})}(S, \mathcal{C}) = \text{MEV}_{\{\mathbf{C0}\}}(S, \{\mathbf{C0}\}) = 0$$

This is because the adversary is restricted to calling **C0**, but the transaction would revert since the **require** condition in **C0** is violated. \diamond

Example A.4. To illustrate the need of the assumption (2) in Lemma A.4, consider the contracts:

```
contract C0 { f() { transfer(M, 1:T) } }
contract C1 { f() { C0.f(); } }
```

Let $\mathcal{M} = \{\mathbf{M}\}$, $\mathcal{C} = \{\mathbf{CO}\}$, $\mathcal{D} = \{\mathbf{C1}\}$, and let:

$$S = \mathbf{M}[0:\mathbf{T}] \mid \mathbf{CO}[1:\mathbf{T}] \mid \mathbf{C1}[0:\mathbf{T}]$$

Let $\vec{X} \in \kappa_{\mathcal{D}}(\mathbf{M})$ be the following sequence of transactions:

$$\vec{X} = \mathbf{M}:\mathbf{C1.f}()$$

By executing \vec{X} in S , we have that:

$$S \xrightarrow{\mathbf{M}:\mathbf{C1.f}()} \mathbf{M}[1:\mathbf{T}] \mid \mathbf{CO}[0:\mathbf{T}] \mid \mathbf{C1}[0:\mathbf{T}]$$

Since there are no tokens left in \mathcal{C} , \vec{X} clearly maximises the loss of \mathcal{C} , hence:

$$\text{MEV}_{\mathcal{D}}(S, \mathcal{C}) = 1 \cdot \mathbf{\$1_T}$$

We have that $\mathcal{C}' = \text{deps}(\{\mathbf{CO}\}) \cap \text{deps}(\{\mathbf{C1}\} \setminus \text{deps}(\{\mathbf{CO}\})) = \{\mathbf{CO}\} \not\subseteq \mathcal{D}$, thus violating condition (2). We have that:

$$\text{MEV}_{\mathcal{D} \cap \text{deps}(\mathcal{C})}(S, \mathcal{C}) = \text{MEV}_{\{\mathbf{C1}\} \cap \{\mathbf{CO}\}}(S, \{\mathbf{CO}\}) = \text{MEV}_{\emptyset}(S, \{\mathbf{CO}\}) = 0$$

Therefore, MEV preservation does not hold. \diamond

Proof of Theorem 3

We show the following two equalities, which imply the thesis:

$$\text{MEV}(S \mid \Delta, \dagger\Delta) = \text{MEV}(S \mid \Gamma_{\mathcal{M}} \mid \Delta, \dagger\Delta) \quad (6)$$

$$\text{MEV}_{\dagger\Delta}(S \mid \Delta, \dagger\Delta) = \text{MEV}_{\dagger\Delta}(S \mid \Gamma_{\mathcal{M}} \mid \Delta, \dagger\Delta) \quad (7)$$

Observe that (7) follows directly from Lemma A.3, since $S \mid \Delta \preceq S \mid \Gamma_{\mathcal{M}} \mid \Delta$ and $\dagger\Delta \subseteq \dagger(S \mid \Delta) = \dagger S \cup \dagger\Delta$. Note instead that (6) does not follow from Lemma A.3, since to equate $\text{MEV}_{\dagger(S \mid \Gamma_{\mathcal{M}} \mid \Delta)}(S \mid \Delta, \dagger\Delta)$ and $\text{MEV}_{\dagger(S \mid \Gamma_{\mathcal{M}} \mid \Delta)}(S \mid \Gamma_{\mathcal{M}} \mid \Delta, \dagger\Delta)$, the lemma would require $\dagger(S \mid \Gamma_{\mathcal{M}} \mid \Delta) \subseteq \dagger(S \mid \Delta)$, which is false.

In order to prove (6), we pass through two auxiliary results. We start by proving the following equality:

$$\text{MEV}_{\dagger(S \mid \Gamma_{\mathcal{M}} \mid \Delta)}(S \mid \Gamma_{\mathcal{M}} \mid \Delta, \dagger\Delta) = \text{MEV}_{\dagger(S \mid \Delta) \cap \text{deps}(\dagger\Delta)}(S \mid \Gamma_{\mathcal{M}} \mid \Delta, \dagger\Delta) \quad (8)$$

In order to apply Lemma A.4, let:

$$\mathcal{C} = \dagger\Delta \quad \mathcal{D} = \dagger(S \mid \Gamma_{\mathcal{M}} \mid \Delta)$$

and let:

$$\begin{aligned} \mathcal{C}' &= \text{deps}(\mathcal{C}) \cap \text{deps}(\mathcal{D} \setminus \text{deps}(\mathcal{C})) \\ &= \text{deps}(\Delta) \cap \text{deps}(\dagger(S \mid \Gamma_{\mathcal{M}} \mid \Delta) \setminus \text{deps}(\Delta)) \end{aligned}$$

Note that the conditions of Lemma A.4 are satisfied:

- (1) \mathcal{C}' are sender-agnostic, since $\mathcal{C}' \subseteq \text{deps}(\Delta)$ and, by assumption of Theorem 3, the contracts in $\text{deps}(\Delta)$ are sender-agnostic;
- (2) $\mathcal{C}' \subseteq \mathcal{D}$ holds since $\mathcal{C}' \subseteq \text{deps}(\Delta) \subseteq \mathcal{D}$;
- (3) Since the state $S \mid \Delta$ is well-formed by assumption, then $\text{deps}(\Delta) \subseteq \dagger(S \mid \Delta)$, and so we have that:

$$\begin{aligned} \text{deps}(\mathcal{D}) \cap \text{deps}(\mathcal{C}) &= \text{deps}(S \mid \Gamma_{\mathcal{M}} \mid \Delta) \cap \text{deps}(\Delta) = \text{deps}(\Delta) \\ \text{deps}(\mathcal{D}) \setminus \text{deps}(\mathcal{C}) &= \text{deps}(S \mid \Gamma_{\mathcal{M}} \mid \Delta) \setminus \text{deps}(\Delta) = \text{deps}(S \mid \Gamma_{\mathcal{M}}) \setminus \text{deps}(\Delta) \\ &\subseteq \dagger(S \mid \Gamma_{\mathcal{M}}) \setminus \text{deps}(\Delta) \end{aligned}$$

Since $S \mid \Gamma_{\mathcal{M}} \mid \Delta$ is well-formed and $\text{deps}(\Delta)$ and $\dagger(S \mid \Gamma_{\mathcal{M}}) \setminus \text{deps}(\Delta)$ are disjoint, then Condition (ii) of Theorem 3 ensures that these sets are token independent.

Therefore, by Lemma A.4 it follows that:

$$\text{MEV}_{\dagger(S \mid \Gamma_{\mathcal{M}} \mid \Delta)}(S \mid \Gamma_{\mathcal{M}} \mid \Delta, \dagger\Delta) = \text{MEV}_{\dagger(S \mid \Gamma_{\mathcal{M}} \mid \Delta) \cap \text{deps}(\dagger\Delta)}(S \mid \Gamma_{\mathcal{M}} \mid \Delta, \dagger\Delta)$$

To obtain (8), just note that, since $\text{deps}(\dagger\Delta) \subseteq \dagger(S \mid \Delta)$:

$$\dagger(S \mid \Gamma_{\mathcal{M}} \mid \Delta) \cap \text{deps}(\dagger\Delta) = \dagger(S \mid \Delta) \cap \text{deps}(\dagger\Delta)$$

which is equal to $\text{deps}(\dagger\Delta)$.

The second auxiliary result is the equality:

$$\text{MEV}_{\dagger(S \mid \Delta)}(S \mid \Gamma_{\mathcal{M}} \mid \Delta, \dagger\Delta) = \text{MEV}_{\dagger(S \mid \Delta) \cap \text{deps}(\dagger\Delta)}(S \mid \Gamma_{\mathcal{M}} \mid \Delta, \dagger\Delta) \quad (9)$$

This time, in order to apply Lemma A.4 we let:

$$\mathcal{C} = \dagger\Delta \quad \mathcal{D} = \dagger(S \mid \Delta) \quad \mathcal{C}' = \text{deps}(\Delta) \cap \text{deps}(\mathcal{D} \setminus \text{deps}(\mathcal{C}))$$

In order to apply Lemma A.4, let us first compute:

$$\begin{aligned} \mathcal{C}' &= \text{deps}(\mathcal{C}) \cap \text{deps}(\mathcal{D} \setminus \text{deps}(\mathcal{C})) \\ &= \text{deps}(\Delta) \cap \text{deps}(\dagger(S \mid \Delta) \setminus \text{deps}(\Delta)) \end{aligned}$$

Again, note that the assumptions of Lemma A.4 are satisfied:

- (1) \mathcal{C}' are sender-agnostic, since $\mathcal{C}' \subseteq \text{deps}(\Delta)$ and assumption (i);
- (2) $\mathcal{C}' \subseteq \mathcal{D}$ holds since $\mathcal{C}' \subseteq \text{deps}(\Delta) \subseteq \mathcal{D}$;
- (3) Since the state $S \mid \Delta$ is well-formed by assumption, then $\text{deps}(\Delta) \subseteq \dagger(S \mid \Delta)$, and so we have that:

$$\begin{aligned} \text{deps}(\mathcal{D}) \cap \text{deps}(\mathcal{C}) &= \text{deps}(S \mid \Delta) \cap \text{deps}(\Delta) = \text{deps}(\Delta) \\ \text{deps}(\mathcal{D}) \setminus \text{deps}(\mathcal{C}) &= \text{deps}(S \mid \Delta) \setminus \text{deps}(\Delta) = \text{deps}(S) \setminus \text{deps}(\Delta) \end{aligned}$$

Condition (ii) ensures that these sets are token independent.

Therefore, Lemma A.4 gives the equality (8).

Now, by putting together (8) and (9), we obtain:

$$\text{MEV}_{\dagger(S|\Gamma_{\mathcal{M}}|\Delta)}(S | \Gamma_{\mathcal{M}} | \Delta, \dagger\Delta) = \text{MEV}_{\dagger(S|\Delta)}(S | \Gamma_{\mathcal{M}} | \Delta, \dagger\Delta) \quad (10)$$

Now we can prove (6) by observing the following chain of equalities:

$$\begin{array}{ccc} \text{MEV}_{\dagger(S|\Delta)}(S | \Gamma_{\mathcal{M}} | \Delta, \dagger\Delta) & \xrightarrow{\text{by (10)}} & \text{MEV}_{\dagger(S|\Gamma_{\mathcal{M}}|\Delta)}(S | \Gamma_{\mathcal{M}} | \Delta, \dagger\Delta) \\ \parallel \text{by Lemma A.3} & & \parallel \\ \text{MEV}_{\dagger(S|\Delta)}(S | \Delta, \dagger\Delta) & & \text{MEV}_{\dagger(S|\Gamma_{\mathcal{M}}|\Delta)}(S | \Gamma_{\mathcal{M}} | \Delta, \dagger\Delta) \\ \parallel \text{by Lemma A.1(4)} & & \parallel \text{by Lemma A.1(4)} \\ \text{MEV}(S | \Delta, \dagger\Delta) & & \text{MEV}(S | \Gamma_{\mathcal{M}} | \Delta, \dagger\Delta) \end{array}$$

Now, the thesis directly follows from Equations (6) and (7). \square

B Proofs: use cases

AMM/Bet (Example 5)

Consider the starting state:

$$\begin{aligned} S &= \mathbf{M}[m: \text{ETH}] | \mathbf{AMM}[r_0: \text{ETH}, r_1: \mathbf{T}] | \text{block.num} = d - k | \dots \\ \Delta &= \mathbf{Bet}[b: \text{ETH}, \text{owner} = \mathbf{A}, \text{tok} = \mathbf{T}, \text{rate} = r, \text{deadline} = d] \end{aligned}$$

When \mathbf{M} is allowed to manipulate the \mathbf{AMM} , she can inflate the exchange rate of ETH , provided that she possesses sufficient funds. Formally, if \mathbf{M} swaps $x: \text{ETH}$ for $y: \mathbf{T}$, then according to the criterion specified in $\mathbf{Bet.win}()$, the winner receives an amount $\lfloor 2bp \rfloor$ only if $\mathbf{AMM.getRate}(\text{ETH}) = r_0 + x / r_1 - y \geq p \cdot r$. Assuming that \mathbf{M} enters the bet only when she can choose x sufficiently high to satisfy this condition, and for $p \geq 1/2$ (since a smaller proportion makes the bet irrational for her), she fires the following sequence of transactions: where, in the swap transaction, $x = m - b \geq 0$ is the number of ETH units sent to the \mathbf{AMM} , $y = \lfloor x r_1 / r_0 + x \rfloor$ is the number of \mathbf{T} units received, and the value that \mathbf{M} bets on is $p = r_0 + x / r(r_1 - y)$:

$$\begin{array}{l} S | \Delta \xrightarrow{\mathbf{M:Bet.bet}(\mathbf{M} \text{ pays } b: \text{ETH}, p)} \mathbf{AMM}[r_0: \text{ETH}, r_1: \mathbf{T}] | \mathbf{Bet}[2b: \text{ETH}, \text{potShare} = p, \dots] | \dots \\ \xrightarrow{\mathbf{M:AMM.swap}(\mathbf{M} \text{ pays } x: \text{ETH}, 0)} \mathbf{AMM}[r_0 + x: \text{ETH}, r_1 - y: \mathbf{T}] | \mathbf{Bet}[2b: \text{ETH}, \dots] | \dots \\ \xrightarrow{\mathbf{M:Bet.win}()} \mathbf{AMM}[r_0 + x: \text{ETH}, r_1 - y: \mathbf{T}] | \mathbf{Bet}[2b - \lfloor 2bp \rfloor: \text{ETH}, \dots] | \dots \\ \xrightarrow{\mathbf{M:AMM.swap}(\mathbf{M} \text{ pays } y: \mathbf{T}, 0)} \mathbf{AMM}[r_0: \text{ETH}, r_1: \mathbf{T}] | \mathbf{Bet}[2b - \lfloor 2bp \rfloor: \text{ETH}, \dots] | \dots \end{array}$$

By Equation (2) we have:

$$\begin{aligned}
\text{MEV}(S \mid \Delta, \{\text{Bet}\}) &= b - (2b - \lfloor 2bp \rfloor) = \lfloor 2bp \rfloor - b \leq 2bp - b = \frac{2b(r_0 + x)}{r(r_1 - y)} - b \\
&= \frac{2b(r_0 + x)}{r \left(r_1 - \left\lfloor \frac{xr_1}{r_0 + x} \right\rfloor \right)} - b \leq \frac{2b(r_0 + x)}{r \left(r_1 - \frac{xr_1}{r_0 + x} \right)} - b \\
&= \frac{2b(r_0 + x)^2}{rr_0r_1} - b = \left(\frac{2(r_0 + m - b)^2}{rr_0r_1} - 1 \right) b
\end{aligned}$$

Whereas, if **M** was restricted to interact with **Bet** only, there are two cases: if $\text{AMM.getRate} = r_0/r_1 \geq p \cdot r$, then **M** wins the bet. Otherwise, she loses (and, therefore, **Bet** does not suffer an economic loss). Even in this case, **M** enters the bet only for $p \geq 1/2$. Therefore Equation (3) gives us:

$$\begin{aligned}
\text{MEV}_{\{\text{Bet}\}}(S \mid \Delta, \{\text{Bet}\}) &= \begin{cases} b - \left(2b - \left\lfloor \frac{2br_0}{rr_1} \right\rfloor \right) & \text{if } \frac{r_0}{rr_1} \geq 1/2 \\ 0 & \text{otherwise} \end{cases} \\
&= \begin{cases} \left\lfloor \frac{2br_0}{rr_1} \right\rfloor - b & \text{if } \frac{r_0}{rr_1} \geq 1/2 \\ 0 & \text{otherwise} \end{cases} \\
&> \begin{cases} \frac{2br_0}{rr_1} - b - 1 & \text{if } \frac{r_0}{rr_1} \geq 1/2 \\ 0 & \text{otherwise} \end{cases}
\end{aligned}$$

Hence MEV interference is estimated through Definition 1 as follows:

$$\begin{aligned}
\mathcal{J}(S \rightsquigarrow \Delta) &< \begin{cases} 1 - \frac{\frac{2br_0}{rr_1} - b - 1}{\left(\frac{2(r_0 + m - b)^2}{rr_0r_1} - 1 \right) b} & \text{if } \frac{r_0}{rr_1} \geq 1/2 \\ 1 & \text{otherwise} \end{cases} \\
&= \begin{cases} 1 - \frac{(2br_0 - brr_1 - rr_1)r_0}{b(2(r_0 + m - b)^2 - rr_0r_1)} & \text{if } \frac{r_0}{rr_1} \geq 1/2 \\ 1 & \text{otherwise} \end{cases} \\
&= \begin{cases} 1 - \frac{2br_0^2 - rr_0r_1(b+1)}{2b(r_0 + m - b)^2 - brr_0r_1} & \text{if } \frac{r_0}{rr_1} \geq 1/2 \\ 1 & \text{otherwise} \end{cases} \quad \diamond
\end{aligned}$$

AMM/Lending Pool (Example 6)

For simplicity, we make the following assumptions: (i) $\$1_{\text{ETH}} = 1 = \1_{T} , (ii) the **AMM** is balanced, (iii) **M** has not deposited or borrowed tokens from the **LP** yet. (iv) the **LP** has sufficient reserves of **T** to satisfy any borrow request. Note also that our simplified **LP** contract only offers two functions, **deposit** and **borrow**. Calling **deposit** does not extract tokens from the **LP**, so the only action through which **M** could cause a loss to the **LP** is **borrow**.

Consider the following blockchain state:

$$S = \mathbf{M}[n:\text{ETH}] \mid \mathbf{AMM}[r:\text{ETH}, r:\text{T}] \quad \Delta = \mathbf{LP}[a:\text{ETH}, b:\text{T}, \text{Cmin} = C_{\min}, \dots]$$

We start by estimating the unrestricted local MEV, i.e. $\text{MEV}(S \mid \Delta, \{\mathbf{LP}\})$. When \mathbf{M} can interact with the \mathbf{AMM} , she can maximize the loss caused to \mathbf{LP} by maximizing her loan amount, or in other words, by inflating her collateralization ratio. There is only one way to do so: by depositing a portion of her ETH to the \mathbf{LP} and by inflating the exchange rate of ETH provided by the \mathbf{AMM} . To this purpose, \mathbf{M} partitions its funds as follows:

- $x:\text{ETH}$ to perform a swap in the \mathbf{AMM} in exchange for $y:\text{T}$, where $y = xr/r+x$
- $(n-x):\text{ETH}$ to deposit in the \mathbf{LP}

We denote by $t(x)$ the number of units of token T that \mathbf{M} can borrow from the \mathbf{LP} as a function of x .

We first note that in order to satisfy the **require** constraint within **borrow** function of \mathbf{LP} , \mathbf{M} must be over-collateralized in the new \mathbf{LP} state. Recall that the collateralization of a user is given by the ratio between the value of her minted tokens and that of her debts. Regarding \mathbf{M} , the value v_{minted} of her minted tokens and the value v_{debt} of her debts in the new state are given by:

$$\begin{aligned} v_{\text{minted}} &= (n-x) \cdot \frac{\mathbf{AMM.getRate}(\text{ETH})}{\mathbf{AMM.getRate}(\text{T})} = (n-x) \cdot \frac{r-y}{r+x} \\ v_{\text{debt}} &= t(x) \cdot \frac{\mathbf{AMM.getRate}(\text{T})}{\mathbf{AMM.getRate}(\text{ETH})} = t(x) \cdot \frac{r+x}{r-y} \end{aligned}$$

Therefore, \mathbf{M} is over-collateralized, and so her call to **borrow** does not revert, if:

$$\frac{v_{\text{minted}}}{v_{\text{debt}}} = \frac{(n-x)(r+x)^2}{t(x)(r-y)^2} \geq C_{\min}$$

This gives us the maximum value of $t(x)$ that \mathbf{M} can choose, which is:

$$t(x) = \frac{(n-x)(r+x)^2}{C_{\min}(r-y)^2}$$

To find the value of x that maximizes $t(x)$, we study the function $t(x)$ that gives the loan amount as a function of the deposited amount x , subject to the constraint $0 \leq x \leq n$. Since we working with real-valued amounts, we have that $t(x)$ is continuous. Thus, we compute its derivative w.r.t. x and set it to 0:

$$\frac{dt(x)}{dx} = \frac{d}{dx} \left(\frac{(n-x)(r+x)^4}{C_{\min}r^4} \right) = \frac{4(n-x)(r+x)^3 - (r+x)^4}{C_{\min}r^4} = 0$$

Since $r+x > 0$, we can simplify the above as:

$$4(n-x) = r+x \quad \text{if } 0 \leq x \leq n$$

Therefore, the x that maximizes $t(x)$ is given by:

$$x = \begin{cases} \frac{4n-r}{5} & \text{if } 4n \geq r \\ 0 & \text{otherwise} \end{cases}$$

In other words, when $4n < r$, **M** does not need to interact with the **AMM** to maximize her borrowing capacity.

We can check that $x = \frac{4n-r}{5}$ maximizes $t(x)$ by performing the double derivative test. We compute the double derivative of $t(x)$ w.r.t x , plugging in $x = \frac{4n-r}{5}$, and check if it is < 0 . Accordingly:

$$\begin{aligned} \frac{d^2 t(x)}{dx^2} &= \frac{d}{dx} \left(\frac{4(n-x)(r+x)^3 - (r+x)^4}{C_{min} \cdot r^4} \right) \\ &= \frac{12(n-x)(r+x)^2 - 4(r+x)^3 - 4(r+x)^3}{C_{min} \cdot r^4} \\ &= \frac{12(n-x)(r+x)^2 - 8(r+x)^3}{C_{min} \cdot r^4} \end{aligned}$$

Substituting $4(n-x) = r+x$ we get:

$$\frac{d^2 t(x)}{dx^2} = \frac{3(r+x)^3 - 8(r+x)^3}{C_{min} \cdot r^4} = -\frac{5(r+x)^3}{C_{min} \cdot r^4} < 0$$

As a result, **M** fires the following sequence of transactions with a loan amount $t = (n-x)(r+x)^2 / C_{min}(r-y)^2$ and the amount received on swap $y = xr/r+x$:

$$\begin{array}{l} S \mid \Delta \xrightarrow{\text{M:LP.deposit(M pays } (n-x):\text{ETH})} \text{AMM}[r:\text{ETH}, r:\text{T}] \mid \text{LP}[a+n-x:\text{ETH}, b:\text{T}, \dots] \mid \dots \\ \xrightarrow{\text{M:AMM.swap(M pays } x:\text{ETH}, 0)} \text{AMM}[r+x:\text{ETH}, r-y:\text{T}] \mid \text{LP}[a+n-x:\text{ETH}, b:\text{T}, \dots] \mid \dots \\ \xrightarrow{\text{M:LP.borrow}(t,\text{T})} \text{AMM}[r+x:\text{ETH}, r-y:\text{T}] \mid \text{LP}[a+n-x:\text{ETH}, b-t:\text{T}, \dots] \mid \dots \\ \xrightarrow{\text{M:AMM.swap(M pays } y:\text{T}, 0)} \text{AMM}[r:\text{ETH}, r:\text{T}] \mid \text{LP}[a+n-x:\text{ETH}, b-t:\text{T}, \dots] \mid \dots \end{array}$$

By Equation (2) we get:

$$\begin{aligned}
\text{MEV}(S \mid \Delta, \{\text{LP}\}) &= t + x - n = \frac{(n-x)(r+x)^2}{C_{\min}(r-y)^2} + x - n \\
&= (n-x) \left(\frac{(r+x)^2}{C_{\min}(r-\frac{xr}{r+x})^2} - 1 \right) \\
&= (n-x) \left(\frac{(r+x)^4}{r^4 C_{\min}} - 1 \right) \\
&= \begin{cases} \left(n - \frac{4n-r}{5} \right) \left(\frac{(r+\frac{4n-r}{5})^4}{r^4 C_{\min}} - 1 \right) & \text{if } 4n \geq r \\ n \left(\frac{1}{C_{\min}} - 1 \right) & \text{otherwise} \end{cases} \\
&= \begin{cases} \left(\frac{n+r}{5} \right) \left(\frac{(4(n+r))}{r^4 C_{\min}} - 1 \right) & \text{if } 4n \geq r \\ n \left(\frac{1}{C_{\min}} - 1 \right) & \text{otherwise} \end{cases} \\
&= \begin{cases} \left(\frac{n+r}{5} \right) \left(\frac{1}{C_{\min}} \left(\frac{4(n+r)}{5r} \right)^4 - 1 \right) & \text{if } 4n \geq r \\ n \left(\frac{1}{C_{\min}} - 1 \right) & \text{otherwise} \end{cases}
\end{aligned}$$

We note two key aspects of the transaction sequence fired by **M**. Firstly, the ordering of **deposit** and the (initial) **swap** transactions is irrelevant. Hence, they can be interchanged without causing a difference to the loss caused to **LP**. Secondly, firing the (final) **swap**, i.e. de-manipulating the **AMM** only affects the wealth of **M** and not the **LP**. Hence, it does not affect the MEV extractable from **LP**. Nevertheless, we include it in the transaction sequence to reflect the attack execution employed in practice.

We now calculate the restricted local MEV, i.e. $\text{MEV}_{\{\text{LP}\}}(S \mid \Delta, \{\text{LP}\})$. In this case, the only way **M** can maximize her borrowing capacity is by depositing her total available capital to the **LP**. Hence, **M** deposits n :**ETH**. The collateralization of **M** after a call to **borrow** for t' units of **T** is given by:

$$\frac{v_{\text{minted}}}{v_{\text{debt}}} = \frac{n \cdot \text{AMM.getRate}(\text{ETH}) / \text{AMM.getRate}(\text{T})}{t' \cdot \text{AMM.getRate}(\text{T}) / \text{AMM.getRate}(\text{ETH})} = \frac{n \cdot r/r}{t' \cdot r/r} = \frac{n}{t'(x)}$$

Thus, the call to **borrow** does not revert iff:

$$\frac{n}{t'} \geq C_{\min}$$

From this, we obtain that the maximum amount that **M** can borrow is given by:

$$t' = \frac{n}{C_{min}}$$

By Equation (3), we have that:

$$\text{MEV}_{\{\text{LP}\}}(S \mid \Delta, \{\text{LP}\}) = t' - n = \frac{n}{C_{min}} - n = n \left(\frac{1}{C_{min}} - 1 \right)$$

To conclude, we estimate MEV interference through Definition 1 as follows:

$$\begin{aligned} \mathcal{J}(S \rightsquigarrow \Delta) &= \begin{cases} 1 - \frac{n \left(\frac{1}{C_{min}} - 1 \right)}{\left(\frac{n+r}{5} \right) \left(\frac{1}{C_{min}} \left(\frac{4(n+r)}{5r} \right)^4 - 1 \right)} & \text{if } 4n \geq r \\ 1 - \frac{n \left(\frac{1}{C_{min}} - 1 \right)}{n \left(\frac{1}{C_{min}} - 1 \right)} & \text{otherwise} \end{cases} \\ &= \begin{cases} 1 - \frac{n(1-C_{min})}{C_{min}} \cdot \frac{5}{n+r} \cdot \frac{(5r)^4 C_{min}}{(4(n+r))^4 - (5r)^4 C_{min}} & \text{if } 4n \geq r \\ 0 & \text{otherwise} \end{cases} \\ &= \begin{cases} 1 - \frac{5^5 r^4 n(1-C_{min})}{(n+r)(4^4(n+r)^4 - (5r)^4 C_{min})} & \text{if } 4n \geq r \\ 0 & \text{otherwise} \end{cases} \quad \diamond \end{aligned}$$