A quantitative notion of economic security for smart contract compositions

No Author Given

No Institute Given

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 an individual component 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 lead to economic losses of the overall composition. We study the fundamental properties of this notion and apply it to assess the security of notable smart contract compositions. In particular, we analyse under-collateralized loan attacks in systems composed 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 [13,14].

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 [11,19,5,18,1].

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" paper [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 [9]. 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 $\mathrm{MEV}(S \mid \Delta) \leq (1+\varepsilon)\,\mathrm{MEV}(S)$: 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, since interactions with S are immaterial.

In a different security setting, a similar intuition was the basis of Goguen and Meseguer' non-interference [10], 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 [12,22] are also based on the idea of non-interference, but replacing MEV with an explicit tagging of contract variables into 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 such a qualitative evaluation is sufficient when a composition is deemed secure, in case it is not it fails to give any meaningful estimate of the *degree* of interference. For example, when in the above-mentioned (insecure) composition between a lending protocol and a DEX, 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 $\mathcal{I}(S \rightsquigarrow \Delta)$, 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, $\mathcal{I}(S \rightsquigarrow \Delta)$ increases when S is extended with contracts that are not in the dependencies of Δ (Theorem 1); $\mathcal{I}(S \rightsquigarrow \Delta)$ does not depend on the token balances of users except adversaries (Theorem 2); $\mathcal{I}(S \rightsquigarrow \Delta)$ 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.

$\mathtt{A}, \mathtt{B}$	User accounts	$\mathcal{A}, \mathcal{B}$	Sets of [user|contract] accounts
$\mathtt{C}, \mathtt{D}$	Contract accounts	$\mathcal{C}, \mathcal{D}$	Sets of contract accounts
$\mathtt{T}, \mathtt{T}'$	Token types	$\$\mathbf{1}_{\mathtt{T}}$	Price of $\mathtt{T}$
$\mathtt{X}, \mathtt{X}'$	Transactions	$\vec{\mathtt{X}}$	Sequence of transactions
S, S'	Blockchain states	$\$_{\mathcal{C}}(S)$	Wealth of contracts $\mathcal{C}$ in S
W, W'	Wallet states	$\mathit{deps}(\mathcal{C})$	Dependencies of contracts $\mathcal{C}$
Γ, Δ	Contract states	$\dagger\Gamma$	Contract accounts in Γ

2 Smart contracts model

We consider a contract model inspired by account-based platforms *à la* Ethereum. The basic building blocks of our model are a set $\mathbb{T}$ of *token types* ($\mathtt{T}, \mathtt{T}', \ldots$), representing the native crypto-assets (e.g., ETH), and a set $\mathbb{A}$ of *accounts*. Accounts can be *user accounts* $\mathtt{A}, \mathtt{B}, \ldots \in \mathbb{A}_u$ (representing the so-called *externally owned accounts* in Ethereum) and *contract accounts* $\mathtt{C}, \mathtt{D}, \ldots \in \mathbb{A}_c$.

The state of a user account is a map $w \in \mathbb{T} \to \mathbb{N}$ 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* S is a map from accounts to their states. We write an account state in square brackets, wherein we denote by $n\colon\mathtt{T}$ a balance of n units of token $\mathtt{T}$ in the wallet, and with $\mathtt{x} = v$ an association of value v to the storage variable $\mathtt{x}$. For example, $\mathtt{C}[1\colon\mathtt{T}, \mathtt{owner} = \mathtt{A}]$ represents a state where the contract $\mathtt{C}$ stores 1 unit of $\mathtt{T}$, and the variable $\mathtt{owner}$ contains the address $\mathtt{A}$. We write a blockchain state as the composition of its account states, using the symbol $\mid$ as a separator. For example, $S = \mathtt{A}[1\colon\mathtt{T}, 2\colon\mathtt{ETH}] \mid \mathtt{C}[1\colon\mathtt{T}, \mathtt{owner} = \mathtt{A}]$ is a state composed by a user account and a contract account.

We abstractly model contracts as the transitions they induce on blockchain states, just assuming that contracts have an associated set of *functions*, which can be called by *transactions* sent by users. More precisely, we assume a deterministic transition relation $\to$ between blockchain states, where state transitions are triggered by transactions $\mathtt{X}, \mathtt{X}', \ldots$. Similarly to Solidity, we assume that functions 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 contracts (possibly transferring tokens), (v) return values to the caller. Functions can only manipulate tokens as described above: in particular, they cannot drain tokens from other accounts, nor can they mint or burn tokens. In our use cases in Section 4, we will instantiate this abstract model using a contract language inspired by Solidity. A transaction is a call to a contract function, written $\mathtt{A}\colon\mathtt{C}.\mathtt{f}(\mathtt{args})$, where $\mathtt{A}$ is the user signing the transaction, $\mathtt{C}$ is the called contract, $\mathtt{f}$ is the called function, and $\mathtt{args}$ is the list of actual parameters. Parameters can also include transfers of tokens from $\mathtt{A}$ to $\mathtt{C}$, written $\mathtt{A}\ \mathtt{pays}\ n\colon\mathtt{T}$. Invalid transactions are reverted (i.e., they do not update the blockchain state).

We assume that a contract $\mathtt{C}$ can call a function of a contract $\mathtt{D}$ only if $\mathtt{D}$ was deployed before $\mathtt{C}$. Formally, defining $\mathtt{C} \prec \mathtt{D}$ (read: "$\mathtt{C}$ *is called by* $\mathtt{D}$") when some function in $\mathtt{D}$ calls some function in $\mathtt{C}$, we require that the transitive and reflexive closure $\sqsubseteq$ of $\prec$ is a partial order. We define the *dependencies* of a contract $\mathtt{C}$ as $\mathit{deps}(\mathtt{C}) = \{\mathtt{C}' \mid \mathtt{C}' \sqsubseteq \mathtt{C}\}$, and extend this notion to *sets* of contracts $\mathcal{C} = \{\mathtt{C}_1, \ldots, \mathtt{C}_n\}$. We assume that blockchain states S enjoy the following conditions: (i) S contains all its dependencies, i.e. if $\mathtt{C}$ is a contract in S, then also the contracts $\mathit{deps}(\mathtt{C})$ are in S; (ii) S contains *finite tokens*. All states mentioned in our results are assumed to enjoy these well-formedness assumption.[1] 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. Given $\mathtt{X} = \mathtt{A}\colon\mathtt{C}.\mathtt{f}(\mathtt{args})$, we write $\mathit{callee}(\mathtt{X})$ for the target contract $\mathtt{C}$.

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 blockchain state $S \mid \Delta$ 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 $\mathtt{C} \in \mathcal{C}$ 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 $\$\mathbf{1}_{\mathtt{T}}$ the price of a token type $\mathtt{T}$, the wealth of a single contract state $\mathtt{C}[w, \sigma]$ is given by $\sum_{\mathtt{T}} w(\mathtt{T}) \cdot \$\mathbf{1}_{\mathtt{T}}$, i.e. the summation, for all token types $\mathtt{T}$, of the number of tokens $\mathtt{T}$ in the wallet of $\mathtt{C}$, times the price $\mathtt{T}$.[2] By extending this to all contracts in $\mathcal{C}$, we obtain the following general definition of wealth:

$$\$_{\mathcal{C}}(S) \;=\; \sum_{\mathtt{C} \in \mathcal{C}, \mathtt{T}} \mathit{fst}(\Gamma(\mathtt{C}))(\mathtt{T}) \cdot \$\mathbf{1}_{\mathtt{T}} \qquad (1)$$

[1] 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.

[2] Here we implicitly assume that the of *native* crypto-assets are constant, since they do not depend on the blockchain state. We discuss this limitation in Section 5.

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 contract $\mathcal{C}$ in a blockchain state S, denoted by $\mathrm{MEV}(S, \mathcal{C})$, 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 $\vec{\mathsf{X}}$ of transactions in $\kappa(\mathcal{M})$. In symbols:

$$\mathrm{MEV}(S, \mathcal{C}) = \max \left\{ \$_{\mathcal{C}}(S) - \$_{\mathcal{C}}(S') \;\middle|\; \vec{\mathsf{X}} \in \kappa(\mathcal{M})^*,\ S \xrightarrow{\vec{\mathsf{X}}} S' \right\} \tag{2}$$

Note that in $\mathrm{MEV}(S, \mathcal{C})$, adversaries are allowed to call any contract in S, and in particular the dependencies of $\mathcal{C}$. Technically, this follows from the fact that $\kappa(\mathcal{M})$ does not pose any restriction on 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 $\kappa_{\mathcal{D}}(\mathcal{M}) = \{\mathsf{X} \in \kappa(\mathcal{M}) \mid \mathit{callee}(\mathsf{X}) \in \mathcal{D}\}$ the set of transactions craftable by $\mathcal{M}$ and targeting contracts in $\mathcal{D}$. We define:

$$\mathrm{MEV}_{\mathcal{D}}(S, \mathcal{C}) = \max \left\{ \$_{\mathcal{C}}(S) - \$_{\mathcal{C}}(S') \;\middle|\; \vec{\mathsf{X}} \in \kappa_{\mathcal{D}}(\mathcal{M})^*,\ S \xrightarrow{\vec{\mathsf{X}}} S' \right\} \tag{3}$$

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, study its theoretical properties, and apply it to analyze some archetypal compositions. Before doing that, we motivate our definition by discussing a few intuitions about its properties.

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:

- $\mathrm{MEV}(S \mid \Delta, \dagger\Delta)$, the maximal loss of the contracts in Δ, where adversaries are able to send transactions to *any* contract in $S \mid \Delta$;
- $\mathrm{MEV}_{\dagger\Delta}(S \mid \Delta, \dagger\Delta)$, 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"*).

We will call our security notion *MEV interference*, and denote with $\mathcal{I}(S \rightsquigarrow \Delta)$ the MEV interference caused by a blockchain state S to a set of contract states Δ.

4.1 Intuitions

We now enumerate and discuss the driving intuitions behind our definition.

Intuition 1 *If Δ has zero wealth, then $\mathcal{I}(S \rightsquigarrow \Delta)$ must be zero.*

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, according to Intuition 1, we want $\mathcal{I}(S \rightsquigarrow \Delta) = 0$ whenever Δ has zero wealth, not ruling out the possibility of having $\mathcal{I}(S \rightsquigarrow \Delta') > 0$ in a state Δ' where the contracts in Δ have been funded.

Intuition 2 *$\mathcal{I}(S \rightsquigarrow \Delta)$ 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.*

For example, consider an arbitrary S where we want to deploy an airdrop contract Δ that transfers tokens from its balance to any entity who requires them (see Listing 1.1). In this setting, 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, we require that the MEV interference from S to Δ is zero. Note that this intuition corresponds to the requirement that a zero *quantitative* MEV interference is equivalent to the notion of *qualitative* MEV non-interference introduced in [6].

Intuition 3 *$\mathcal{I}(S \rightsquigarrow \Delta)$ should not decrease if we extend S with contracts that are not dependencies of Δ.*

For example, consider a state S where we want to deploy new contracts Δ, with an interference estimated as $\mathcal{I}(S \rightsquigarrow \Delta)$. 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 (actually, we would violate the well-formedness assumption of Section 2). Now, the interference $\mathcal{I}(S \mid \Gamma \rightsquigarrow \Delta)$ could either be equal to $\mathcal{I}(S \rightsquigarrow \Delta)$, or possibly increase when the adversary can drain tokens from Γ to inflict more loss to Δ. Intuition 3 states that, in any case, the interference should not decrease.

Intuition 4 *$\mathcal{I}(S \rightsquigarrow \Delta)$ should be independent of the users' wallets in S, except for those belonging to adversaries.*

Here the intuition is that the adversary does not have any control on 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 the them. We discuss this in Section 5.

Intuition 5 *$\mathcal{I}(S \rightsquigarrow \Delta)$ should have a maximum, 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.*

For example, consider a state S that contains an airdrop contract releasing one unit of a token type `T`. We want to deploy a new contract Δ that, upon the payment of $1{:}\,\texttt{T}$, releases all its balance of $n{:}\,\texttt{ETH}$. Assume that the adversary has no tokens of type `T`, so that she needs to extract $1{:}\,\texttt{T}$ 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.:

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

then we would obtain that $\mathcal{I}(S \rightsquigarrow \Delta) = n \cdot \$\mathbf{1}_{\texttt{ETH}}$, 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. Assuming to constrain interference in the range $[0, 1]$, the extreme 0 would represent the case where the context is not useful to extract MEV from Δ (as per Intuition 2), while the extreme 1 would represent the case where *all* the MEV extractable from Δ depends on making the adversary interact with the context.

Armed with our list of desiderata, we define MEV interference. We will see in Section 4.3 that this notion is coherent with all our intuitions. MEV interference measures the fraction of $\mathrm{MEV}(S \mid \Delta, \dagger\Delta)$ that can be extracted without using the dependencies of $\dagger\Delta$.

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:*

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

We observe that our notion is coherent to MEV non-interference, meaning that $\mathcal{I}(S \rightsquigarrow \Delta) = 0$ if and only if S and Δ are non-interferent according to [6].

4.2 Use cases

We now illustrate our notion through a set of relevant use cases.

Listing 1.1: A simple airdrop contract.

```
contract Airdrop {
  fund(pay x:T) { } // any user can deposit x:T to the contract
  withdraw(n) {     // any user can withdraw any amount of T from AirDrop
    require(balance(T)<=n); // check that the contract has at least n:T
    transfer(sender,n:T);   // transfer n: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(n) {
    fee = (FeeManager.getFee() * n) / 100;  // get feeRate from FeeManager
    transfer(sender, n-fee:T);
    transfer(FeeManager.getOwner(), fee:T); // send fee to owner
  }
}
contract FeeManager {
  constructor() { owner=sender; feeRate=1; }
  getOwner() { return owner; }
  getFee()   { return feeRate; }
  setFee(r)  { require (r>=0 && r<=100); feeRate=r; }
}
```

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

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

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

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 $\Delta = \texttt{AirdropFee}[n{:}\,\texttt{T}]$ in a blockchain state S containing $\texttt{FeeManager}[\texttt{feeRate} = r]$. The unrestricted MEV is $n \cdot \$\mathbf{1}_{\texttt{T}}$, since an adversary can set the fee to 0 by calling $\texttt{FeeManager.setFee}(0)$ and then withdraw the full balance of $n{:}\,\texttt{T}$ from AirdropFee. Instead, the restricted MEV only amounts to $(n - {}^{r \cdot n}/_{100}) \cdot \$\mathbf{1}_{\texttt{T}}$, since the adversary cannot call FeeManager to manipulate the fee rate. Therefore, if $n > 0$:

$$\mathcal{I}(S \rightsquigarrow \Delta) = 1 - \frac{n - {}^{r \cdot n}/_{100}}{n} = \frac{r}{100}$$

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. ◇

Example 3 (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:

$$S = \texttt{M}[n_{\texttt{M}} : \texttt{T}] \mid \texttt{Airdrop}[n_{\texttt{A}} : \texttt{T}]$$
$$\Delta = \texttt{Exchange}[n_{\texttt{E}} : \texttt{ETH}, \texttt{tin} = \texttt{T}, \texttt{tout} = \texttt{ETH}, \texttt{rate} = r, \texttt{owner} = \texttt{A}]$$

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 `rate` is a floating-point number, and arithmetic operations are floored, and that $\$\mathbf{1}_{\texttt{T}} = \$\mathbf{1}_{\texttt{ETH}} = 1$. We evaluate the MEV interference from S to Δ. When the exchange rate is favourable, i.e. the `rate` is greater than 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 $n_{\texttt{A}} : \texttt{T}$ from `Airdrop`, and swapping these tokens through the `Exchange`. More precisely, we have:

$$\mathrm{MEV}_{\{\texttt{Exchange}\}}(S \mid \Delta, \{\texttt{Exchange}\}) = \begin{cases} \lfloor n_{\texttt{M}} \cdot r \rfloor & \text{if } \lfloor n_{\texttt{M}} \cdot r \rfloor < n_{\texttt{E}} \\ n_{\texttt{E}} & \text{otherwise} \end{cases}$$
$$\mathrm{MEV}(S \mid \Delta, \{\texttt{Exchange}\}) = \begin{cases} \lfloor (n_{\texttt{M}} + n_{\texttt{A}}) \cdot r \rfloor & \text{if } \lfloor (n_{\texttt{M}} + n_{\texttt{A}}) \cdot r \rfloor < n_{\texttt{E}} \\ n_{\texttt{E}} & \text{otherwise} \end{cases}$$

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

$$\mathcal{I}(S \rightsquigarrow \Delta) \leq \begin{cases} n_{\texttt{A}} / (n_{\texttt{M}} + n_{\texttt{A}}) & \text{if } \lfloor (n_{\texttt{M}} + n_{\texttt{A}}) \cdot r \rfloor < n_{\texttt{E}} \\ 1 - n_{\texttt{M}} \cdot r / n_{\texttt{E}} & \text{if } \lfloor (n_{\texttt{M}} + n_{\texttt{A}}) \cdot r \rfloor \geq n_{\texttt{E}} > \lfloor n_{\texttt{M}} \cdot r \rfloor \\ 0 & \text{otherwise} \end{cases}$$

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. ◇

Example 4 (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. `Bet` receives the initial pot from its owner upon deployment, and a player must match this amount to enter the bet.

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
  }
}
```

Before the deadline, the player can win a proportion `potShare` of the total pot if the oracle exchange rate exceeds or equals `potShare` times the bet rate. The remaining portion is taken by the owner as a fee. Consider an instance of `Bet` using the Automated Market Maker `AMM` in Listing 1.5 as a price oracle:

$$S = \mathtt{M}[m\colon \mathtt{ETH}] \mid \mathtt{AMM}[r_0\colon \mathtt{ETH}, r_1\colon \mathtt{T}] \mid \mathtt{block.num} = d - k \mid \cdots$$
$$\Delta = \mathtt{Bet}[b\colon \mathtt{ETH}, \mathtt{owner} = \mathtt{A}, \mathtt{tok} = \mathtt{T}, \mathtt{rate} = r, \mathtt{deadline} = d]$$

When `M` is allowed to leverage `Bet`'s dependency, she can manipulate the `AMM` to influence the internal exchange rate. Given `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 $y = \lfloor {xr_1}/{r_0+x} \rfloor$ is the number of `T` units received (we omit `M`'s wallet for brevity):

$$\begin{aligned}
S \mid \Delta &\xrightarrow{\mathtt{M{:}Bet.bet(M\ pays}\ b{:}\mathtt{ETH},p)} && \mathtt{AMM}[r_0\colon \mathtt{ETH}, r_1\colon \mathtt{T}] \mid \mathtt{Bet}[2b\colon \mathtt{ETH}, \mathtt{potShare} = p, \cdots] \mid \cdots \\
&\xrightarrow{\mathtt{M{:}AMM.swap(M\ pays}\ x{:}\mathtt{ETH},0)} && \mathtt{AMM}[r_0 + x\colon \mathtt{ETH}, r_1 - y\colon \mathtt{T}] \mid \mathtt{Bet}[2b\colon \mathtt{ETH}, \cdots] \mid \cdots \\
&\xrightarrow{\mathtt{M{:}Bet.win()}} && \mathtt{AMM}[r_0 + x\colon \mathtt{ETH}, r_1 - y\colon \mathtt{T}] \mid \mathtt{Bet}[2b(1-p)\colon \mathtt{ETH}, \cdots] \mid \cdots \\
&\xrightarrow{\mathtt{M{:}AMM.swap(M\ pays}\ y{:}\mathtt{T},0)} && \mathtt{AMM}[r_0\colon \mathtt{ETH}, r_1\colon \mathtt{T}] \mid \mathtt{Bet}[2b(1-p)\colon \mathtt{ETH}, \cdots] \mid \cdots
\end{aligned}$$

The bet value that maximizes the loss caused to `Bet` depends on `M`'s wealth, and is given by $p = \lfloor {r_0+x}/{r(r_1-y)} \rfloor$. 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:

$$\mathrm{MEV}(S \mid \Delta, \{\mathtt{Bet}\}) \leq \left({2(r_0+m-b)^2}/{rr_0r_1} - 1 \right) b$$

Whereas, if `M` was restricted to interact with `Bet`, she is limited to settle on a lower bet value and Equation (3) gives us:

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

Listing 1.4: A Bet contract.

```
contract Bet_oracle {
  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);
  }
}
```

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

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

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 fortunate to be draining 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). ◇

Example 5 (AMM/LendingPool). The LP contract in Listing 1.6 implements a lending protocol, allowing users to deposit tokens and borrow them only if their *collateralization* is above a certain threshold. The collateralization is the ratio between the value of deposits and that of debits, and is a reflection of the borrowing capacity of a user. LP is parameterized over an oracle that is queried for the token prices. Below we analyse a well-known attack where the underlying oracle is an AMM, which an adversary manipulates to exceed her previously limited borrowing capacity [11,19,5,18,1]. Consider the following instance:

$$S = \mathtt{M}[x + y{:}\,\mathtt{ETH}] \mid \mathtt{AMM}[r_0{:}\,\mathtt{ETH}, r_1{:}\,\mathtt{T}] \quad \Delta = \mathtt{LP}[a{:}\,\mathtt{ETH}, b{:}\,\mathtt{T}, \mathtt{Cmin} = C_{min}, \cdots]$$

When M can interact with the AMM, she deploys her attack strategy as follows: she first uses her capital to inflate the price of T in the AMM. Subsequently, she is capable of borrowing a greater amount of T since the LP now uses an *artificially inflated* price to determine her borrowing capacity. In particular, M fires the

Listing 1.5: A constant-product AMM contract.

```
contract AMM {
  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); // Exchange has enough tout tokens
    transfer(sender, y:out);  // send y units of token tout to sender
  }
  addLiq(pay x0:T0, pay x1:T1) { // add liquidity to the AMM
    require balance(T0)*(balance(T1)-x1) // preserve exchange rate
         == balance(T1)*(balance(T0)-x0);
  }
}
```

following sequence of transactions, with a loan amount $t = \lfloor x(r_0+y)^2/C_{min}(r_1-z)^2 \rfloor$ and the amount received on swap $z = \lfloor yr_1/r_0+y \rfloor$:

$$
\begin{aligned}
S \mid \Delta \xrightarrow{\texttt{M:LP.deposit(M pays } x\texttt{:ETH)}} & \ \texttt{AMM}[r_0\texttt{:ETH}, r_1\texttt{:T}] \mid \texttt{LP}[a+x\texttt{:ETH}, b\texttt{:T}, \cdots] \mid \cdots \\
\xrightarrow{\texttt{M:AMM.swap(M pays } y\texttt{:ETH,0)}} & \ \texttt{AMM}[r_0+y\texttt{:ETH}, r_1-z\texttt{:T}] \mid \texttt{LP}[a+x\texttt{:ETH}, b\texttt{:T}, \cdots] \mid \cdots \\
\xrightarrow{\texttt{M:LP.borrow(}t\texttt{,T)}} & \ \texttt{AMM}[r_0+y\texttt{:ETH}, r_1-z\texttt{:T}] \mid \texttt{LP}[a+x\texttt{:ETH}, b-t\texttt{:T}, \cdots] \mid \cdots \\
\xrightarrow{\texttt{M:AMM.swap(M pays } z\texttt{:T,0)}} & \ \texttt{AMM}[r_0\texttt{:ETH}, r_1\texttt{:T}] \mid \texttt{LP}[a+x\texttt{:ETH}, b-t\texttt{:T}, \cdots] \mid \cdots
\end{aligned}
$$

When the LP has sufficient funds and $\$\mathbf{1}_{\texttt{ETH}} = 1 = \$\mathbf{1}_{\texttt{T}}$, by Equation (2) we get:

$$\text{MEV}(S \mid \Delta, \{\texttt{LP}\}) \leq x\left(\frac{(r_0+y)^4}{C_{min}(r_0 r_1)^2} - 1\right)$$

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

$$\text{MEV}_{\{\texttt{LP}\}}(S \mid \Delta, \{\texttt{LP}\}) > \frac{x r_0^2}{C_{min} r_1^2} - x - 1$$

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

$$\mathcal{I}(S \rightsquigarrow \Delta) < 1 - \frac{x r_0^4 - C_{min} r_0^2 r_1^2 (x+1)}{x((r_0+y)^4 - C_{min} r_0^2 r_1^2)}$$

In accordance to our expectations, the interference is indeed proportional to the attack capital y of the adversary. Naturally, adversaries with a higher manipulation capital have 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. ◇

Listing 1.6: A Lending Pool contract.

```
contract LP {
  constructor(Cmin_) { Cmin = Cmin_; } // collateralization threshold
  collateral(a) { // return a's collateralization
    val_minted = 0;
    for c in minted: val_minted += minted[t][a] * AMM.getRate(t);
    val_debts = 0;
    for c in debts:  val_debts  += debt[t][a] * AMM.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);
  }
}
```

4.3 Properties of MEV interference

We now study the theoretical properties of MEV interference. Because of space constraints, we relegate to Appendix A the proofs of our statements. 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 (coherently with Intuition 5).

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

The following lemma formalises Intuition 1: namely, when the newly deployed contracts Δ have no wealth (i.e., when $\$_{\dagger\Delta}(\Delta) = 0$), then they have no MEV interference from the context.

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

Theorem 1 concretises Intuition 3, stating that widening the blockchain state S potentially increases MEV interference to newly deployed contracts Δ. Formally, this amounts to showing that $\mathcal{I}$ is monotonic with respect to the operation of adding contracts Γ to the context, i.e. $\mathcal{I}(S \rightsquigarrow \Delta) \leq \mathcal{I}(S \mid \Gamma \rightsquigarrow \Delta)$. Note that Definition 1 assumes that the state $S \mid \Delta$ is well-formed: accordingly, the statement of Theorem 1 implicitly assumes that Δ has no dependencies in Γ.

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

The following example shows that the inequality given by Theorem 1 can be 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 6. Consider the following instance of the `Airdrop` being added to a blockchain state S with no contracts, where we want to deploy an `Exchange`:

$$S = \mathtt{M}[0\colon \mathtt{T}] \qquad \Gamma = \mathtt{Airdrop}[1\colon \mathtt{T}, \mathtt{tout} = \mathtt{T}]$$
$$\Delta = \mathtt{Exchange}[100\colon \mathtt{ETH}, \mathtt{tin} = \mathtt{T}, \mathtt{tout} = \mathtt{ETH}, \mathtt{rate} = 10, \mathtt{owner} = \mathtt{B}]$$

By Lemma 1, S does not interfere with Δ. Instead, adding the `Airdrop` to S yields $\mathcal{I}(S \mid \Gamma \rightsquigarrow \Delta) = 1$, since $\mathrm{MEV}_{\{\mathtt{Exchange}\}}(S \mid \Gamma \mid \Delta, \{\mathtt{Exchange}\}) = 0$ while $\mathrm{MEV}(S \mid \Gamma \mid \Delta, \{\mathtt{Exchange}\}) = 10 \cdot \$\mathbf{1}_{\mathtt{ETH}}$. This increase is caused by the ability of `M` to leverage the token dependencies between the newly deployed `Airdrop` contract to extract more MEV from `Exchange` than previously possible. ◇

We observe that, in general, wealthier adversaries are not necessarily capable of causing *greater* interference. Conversely, it is also incorrect to infer that wealthier adversaries cause *less* interference. Therefore, MEV interference does not enjoy monotonicity with respect to the adversary's wealth in either direction. To realise how a wealthier adversary might potentially cause more MEV interference, we revisit Example 4: in the case where `M` does not possess the necessary wealth to produce a volatility in the `AMM`, being richer would have been a benefit. Conversely, we see in Example 3 how richer adversaries need not exploit the context to the same extent as poorer ones, and we have a decrease in interference caused with increasing adversarial wealth.

Theorem 2 concretises Intuition 4, showing that users' wallets are immaterial to the evaluation of MEV interference. Namely, $\mathcal{I}(S \rightsquigarrow \Delta)$ 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* $\operatorname{dom} W_{\mathcal{M}} = \mathcal{M}$*, then* $\mathcal{I}(W_{\mathcal{M}} \mid W \mid \Gamma \rightsquigarrow \Delta) = \mathcal{I}(W_{\mathcal{M}} \mid \Gamma \rightsquigarrow \Delta)$.

We now formalise token independence, which is a prerequisite for Theorem 3. This requires two auxiliary notions: the token types that can be received by Γ in S, denoted $in_S(\Gamma)$, and those that can be sent, denoted by $out_S(\Gamma)$.

Definition 2 (Token independence). *Let* $S = W \mid \Gamma$*. We define:*

- $in_S(\Gamma)$ *as the set of token types* `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* `T` *to some contract in* Γ*.*
- $out_S(\Gamma)$ *as the set of token types* `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* `T` *from some contract in* Γ*.*

We say that contracts in Γ *and* Δ *are* ***token independent*** *in* $S = W \mid \Gamma \mid \Delta$ *when* $in_S(\Gamma) \cap out_S(\Delta) = \emptyset = in_S(\Delta) \cap out_S(\Gamma)$.

Theorem 3 provides sufficient conditions under which an adversary $\mathcal{M}$ attacking the newly deployed contracts in Δ gains no advantage by deploying malicious contracts $\dagger\Gamma_{\mathcal{M}}$ before the attack. Essentially, these conditions guarantee that the interference caused to Δ is preserved when the state S is extended with contracts $\Gamma_{\mathcal{M}}$ satisfying specific conditions, coherently with Intuition 2. Condition (i) requires $\mathit{deps}(\Delta)$ to be sender-agnostic, i.e. its contract methods are unaware of the identity of the `sender`, only being able to use it as a recipient of token transfers. Condition (ii) requires token independence between the (contract) dependencies and the non-dependencies of Δ which could have possibly been exploited by $\mathcal{M}$. Since Definition 1 assumes that states are well-formed, Theorem 3 implicitly assumes that contracts in Δ do not have dependencies in $\Gamma_{\mathcal{M}}$.

Theorem 3. *$\mathcal{I}(S \rightsquigarrow \Delta) = \mathcal{I}(S \mid \Gamma_{\mathcal{M}} \rightsquigarrow \Delta)$ holds if (i) $\mathit{deps}(\Delta)$ are sender-agnostic, and (ii) $\mathit{deps}(\Delta)$ and the non-dependencies of Δ are token independent in $S \mid \Gamma_{\mathcal{M}} \mid \Delta$.*

5 Conclusions

We have proposed a notion of economic security for smart contract compositions, which quantifies the economic loss adversaries can inflict to a contract by targeting its dependencies. We discuss below some limitations and 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 interference is not affected by fluctuations of these prices (while they could depend on the prices provided by DEXes, like in Examples 4 and 5). 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 MEV in [7].

Future work Although a few tools exist for detecting price manipulation attacks in DeFi protocols [21,16,20], and others for estimating MEV opportunities [3,4], none of the existing tools address general economic attacks to smart contract compositions. The technique underlying the detection of price manipulation attacks is *taint analysis*, which aims at identifying potential data flows from low-level to high-level data (in the DeFi setting, flows from to functions that manipulate token prices to functions that transfer tokens). While this technique could be generalised to analyse *qualitative* MEV non-interference, estimating our *quantitative* interference seems to require more advanced techniques. Some inspiration could be drawn from static analysis techniques for information-theoretic interference [8,17,15,2]. We plan to explore this research line in future work.

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 noninterference. 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. 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
9. 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
10. 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
11. 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
12. 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)
13. 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
14. 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

15. 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
16. 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
17. 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
18. 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
19. 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
20. 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
21. 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
22. Yao, S., Ni, H., Myers, A.C., Cecchetti, E.: SCIF: A language for compositional smart contract security (2024), https://arxiv.org/abs/2407.01204

Listing 1.7: Contracts to demonstrate sender-agnosticity.

```
contract C {
  withdraw() { require(sender==C'); transfer(sender,n:T); }
}
contract C' {
  withdraw() { C.withdraw(); transfer(sender,n:T); }
}
```

A Proofs and supplementary material

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* $S \xrightarrow{\vec{X}} S'$.

Equivalently, 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 $S \xrightarrow{\vec{X}} S'$.

Definition A.2 (Sender-agnostic contracts). *A contract is sender-agnostic if the effect of calling each of its methods can be decomposed as follows:*

- *updating the contract states (either directly or through internal calls);*
- *transferring tokens from and to users and contracts;*
- *transferring tokens to its* `sender`.

Further, we require that any call with the same arguments and `origin`*, but distinct* `sender`*, has the same effect, except for the third item where tokens are transferred to the new sender.*

Example A.1. To demonstrate sender-agnosticity, consider the contracts in Listing 1.7. Notice that $\mathcal{C}$ violates sender-agnosticism, since `sender` is used to enable token transfers only to $\mathcal{C}'$.

Lemma A.1 states that solely widening the contract state does not increase the MEV extractable from the targeted contracts. This is because the contracts allowed to be targeted by the adversary, i.e. $\mathcal{D}$, are not increased. Here, for all $S, \mathcal{C}, \mathcal{D} \subseteq \mathbb{A}_c$, writing $\mathrm{MEV}_{\mathcal{D}}(S, \mathcal{C})$ implicitly assumes $\mathcal{D} \subseteq \dagger S$. In other words, the set of contracts callable by the adversary is assumed to be a subset of the contracts in the state. Additionally, whenever contracts in S call contracts in Γ (but not vice-versa), we decompose the contract state as $S \mid \Delta$.

Lemma A.1. $\mathrm{MEV}_{\mathcal{D}}(W \mid \Gamma, \mathcal{C}) = \mathrm{MEV}_{\mathcal{D}}(W \mid \Gamma \mid \Gamma', \mathcal{C})$

Proof. To prove our claim, we take a sequence of transactions $\vec{X} \in \kappa_{\mathcal{D}}(\mathcal{M})^*$ that maximizes the loss $-\gamma_{\mathcal{C}}(W \mid \Gamma, \vec{X})$, and we show that the contract loss stays the same when $\vec{X}$ is executed in $W \mid \Gamma \mid \Gamma'$. We can assume without loss of

generality that $\vec{\mathsf{X}}$ is valid in $W \mid \Gamma$: in particular, it never calls methods of contracts outside of Γ since $\mathcal{D} \subseteq \dagger\Gamma$. Therefore, contracts in Γ' are not affected by $\vec{\mathsf{X}}$, and moreover $\vec{\mathsf{X}}$ is valid in $W \mid \Gamma \mid \Gamma'$. So we have that $W \mid \Gamma \xrightarrow{\vec{\mathsf{X}}} W' \mid \Gamma''$, implies $W \mid \Gamma \mid \Gamma' \xrightarrow{\vec{\mathsf{X}}} W' \mid \Gamma'' \mid \Gamma'$. This holds since there are no internal contract calls from $\dagger\Gamma$ to $\dagger\Gamma'$. Note that this, however, does not restrict contract calls from $\dagger\Gamma'$ to $\dagger\Gamma$. Finally, to prove our claim that the loss stays constant we note that

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

This implies

$$\mathrm{MEV}_{\mathcal{D}}(W \mid \Gamma, \mathcal{C}) \leq \mathrm{MEV}_{\mathcal{D}}(W \mid \Gamma \mid \Gamma', \mathcal{C}) \tag{4}$$

Now, we take a valid sequence of transactions $\vec{\mathsf{Y}} \in \kappa_{\mathcal{D}}(\mathcal{M})^*$ that maximizes the loss $-\gamma_{\mathcal{C}}(W \mid \Gamma \mid \Gamma', \vec{\mathsf{Y}})$. Because $\vec{\mathsf{Y}} \in \kappa_{\mathcal{D}}(\mathcal{M})^*$ consists of transactions targeting contracts in $\mathcal{D} \subseteq \dagger\Gamma$, and since there are no internal contract calls from $\dagger\Gamma$ to $\dagger\Gamma'$, we have that contracts in Γ' are not affected by $\vec{\mathsf{Y}}$. This is because there are no direct or indirect (internal) calls to contracts in Γ'. Because $\vec{\mathsf{Y}}$ is valid in $W \mid \Gamma \mid \Gamma'$, we have that $W \mid \Gamma \mid \Gamma' \xrightarrow{\vec{\mathsf{Y}}} W'' \mid \Gamma''' \mid \Gamma'$. Furthermore, $\vec{\mathsf{Y}}$ is also valid in $W \mid \Gamma$ since it does not consist of any direct/indirect calls to $\dagger\Gamma'$ ($\because \mathcal{D} \subseteq \dagger\Gamma$). So we have that $W \mid \Gamma \mid \Gamma' \xrightarrow{\vec{\mathsf{Y}}} W'' \mid \Gamma''' \mid \Gamma'$ implies $W \mid \Gamma \xrightarrow{\vec{\mathsf{Y}}} W'' \mid \Gamma'''$. And we can trace back the steps to prove that the loss stays constant.

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

This implies

$$\mathrm{MEV}_{\mathcal{D}}(W \mid \Gamma \mid \Gamma', \mathcal{C}) \leq \mathrm{MEV}_{\mathcal{D}}(W \mid \Gamma, \mathcal{C}) \tag{5}$$

(4) and (5) gives

$$\mathrm{MEV}_{\mathcal{D}}(W \mid \Gamma, \mathcal{C}) = \mathrm{MEV}_{\mathcal{D}}(W \mid \Gamma \mid \Gamma', \mathcal{C})$$

□

Proof of Lemma 2

Proof. From Item 2 and Item 5 of Lemma 1 in [6], we have:

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

Now $\$_{\dagger\Delta}(\Delta) = 0$ implies that $\mathrm{MEV}(S \mid \Delta, \dagger\Delta) = 0$, which using Definition 1 gives our thesis. □

Proof of Lemma 1

Proof. For (i), note that $\mathrm{MEV}(S \mid \Delta, \dagger\Delta) = 0$, and so the thesis follows by Definition 1.

For (ii), in the case where $\mathrm{MEV}(S \mid \Delta, \dagger\Delta) = 0$, we have $\mathcal{I}(S \rightsquigarrow \Delta) = 0$ by definition. And in the case where $\mathrm{MEV}(S \mid \Delta, \dagger\Delta) \neq 0$, we have that:

$$\mathcal{I}(\emptyset \rightsquigarrow \Delta) = 1 - \frac{\mathrm{MEV}_{\dagger\Delta}(\emptyset \mid \Delta, \dagger\Delta)}{\mathrm{MEV}(\emptyset \mid \Delta, \dagger\Delta)}$$

By Item 4 of Lemma 1 in [6], $\mathrm{MEV}(\emptyset \mid \Delta, \dagger\Delta) = \mathrm{MEV}_{\dagger\Delta}(\emptyset \mid \Delta, \dagger\Delta)$, this gives $\mathcal{I}(\emptyset \rightsquigarrow \Delta) = 0$.

For (iii), in the case where $\mathrm{MEV}(S \mid \Delta, \dagger\Delta) = 0$, we have $\mathcal{I}(S \rightsquigarrow \Delta) = 0$ by definition. And in the case where $\mathrm{MEV}(S \mid \Delta, \dagger\Delta) \neq 0$, we have that:

$$\begin{aligned}
&0 \leq \mathrm{MEV}_{\dagger\Delta}(S \mid \Delta, \dagger\Delta) \leq \mathrm{MEV}(S \mid \Delta, \dagger\Delta) && \text{by Item 5 and Item 2 in [6]} \\
\implies &0 \leq \frac{\mathrm{MEV}_{\dagger\Delta}(S \mid \Delta, \dagger\Delta)}{\mathrm{MEV}(S \mid \Delta, \dagger\Delta)} \leq 1 \\
\implies &0 \leq 1 - \frac{\mathrm{MEV}_{\dagger\Delta}(S \mid \Delta, \dagger\Delta)}{\mathrm{MEV}(S \mid \Delta, \dagger\Delta)} \leq 1
\end{aligned}$$

which implies $0 \leq \mathcal{I}(S \rightsquigarrow \Delta) \leq 1$. □

Proof of Theorem 1

Proof. By Definition 1, we have two cases:
If $\mathrm{MEV}(S \mid \Delta, \dagger\Delta) = 0$, then $\mathcal{I}(S \rightsquigarrow \Delta) = \mathcal{I}(S \mid \Gamma \rightsquigarrow \Delta) = 0$, hence the thesis holds trivially. Otherwise, if $\mathrm{MEV}(S \mid \Delta, \dagger\Delta) \neq 0$:

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

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

We start by checking that the following inequality holds:

$$\mathrm{MEV}_{\dagger(S|\Delta)}(S \mid \Delta, \dagger\Delta) \leq \mathrm{MEV}_{\dagger(S|\Gamma|\Delta)}(S \mid \Gamma \mid \Delta, \dagger\Delta) \tag{6}$$

We split the proof of (6) into two parts. We first prove that the MEV stays constant when we increase the contract state S:

$$\mathrm{MEV}_{\dagger(S|\Delta)}(S \mid \Delta, \dagger\Delta) = \mathrm{MEV}_{\dagger(S|\Delta)}(S \mid \Gamma \mid \Delta, \dagger\Delta) \tag{7}$$

and then prove that the MEV potentially increases when we increase the number of target contracts available to the adversary $\mathcal{M}$:

$$\mathrm{MEV}_{\dagger(S|\Delta)}(S \mid \Gamma \mid \Delta, \dagger\Delta) \leq \mathrm{MEV}_{\dagger(S|\Gamma|\Delta)}(S \mid \Gamma \mid \Delta, \dagger\Delta) \tag{8}$$

Equation (7) follows from Lemma A.1. Equation (8) follows from Item 2 of Lemma 1 in [6]. So, we have proven that (6) holds. Observe that the following holds by Lemma A.1:

$$\mathrm{MEV}_{\dagger\Delta}(S \mid \Delta, \dagger\Delta) = \mathrm{MEV}_{\dagger\Delta}(S \mid \Gamma \mid \Delta, \dagger\Delta) \tag{9}$$

We know from (6),

$$\mathrm{MEV}_{\dagger(S|\Delta)}(S \mid \Delta, \dagger\Delta) \leq \mathrm{MEV}_{\dagger(S|\Gamma|\Delta)}(S \mid \Gamma \mid \Delta, \dagger\Delta)$$

Taking the reciprocal on both sides gives us

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

Using (9), we get

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

which finally gives us

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

which gives our thesis $\mathcal{I}(S \rightsquigarrow \Delta) \leq \mathcal{I}(S \mid \Gamma \rightsquigarrow \Delta)$. □

Proof of Theorem 2

Proof. From Item 1 of Lemma 2 in [6], we have that:

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

Hence, we have:

$$\mathrm{MEV}_{\dagger(\Gamma|\Delta)}(W_{\mathcal{M}} \mid W \mid \Gamma \mid \Delta, \dagger\Delta) = \mathrm{MEV}_{\dagger(\Gamma|\Delta)}(W_{\mathcal{M}} \mid \Gamma \mid \Delta, \dagger\Delta), \quad \text{and}$$
$$\mathrm{MEV}_{\dagger\Delta}(W_{\mathcal{M}} \mid W \mid \Gamma \mid \Delta, \dagger\Delta) = \mathrm{MEV}_{\dagger\Delta}(W_{\mathcal{M}} \mid \Gamma \mid \Delta, \dagger\Delta)$$

which gives us:

$$\mathcal{I}(W_{\mathcal{M}} \mid W \mid \Gamma \rightsquigarrow \Delta) = \mathcal{I}(W_{\mathcal{M}} \mid \Gamma \rightsquigarrow \Delta)$$

□

Now, we introduce the useful auxiliary notion of *stripping* which will later be used in Theorem 4 and Theorem 3. Formally, given a set of contracts $\mathcal{D}, \mathcal{C} \subseteq S$, we define the stripping of $\mathcal{D}$ w.r.t $\mathcal{C}$, (denoting it with $\mathcal{D}\restriction_{\mathcal{C}}$), as the restriction of $\mathcal{D}$ to the domain $\mathit{deps}(\mathcal{C})$. Theorem 4 gives sufficient conditions under which we can strip $\mathcal{D}$ of all the non-dependencies of $\mathcal{C}$ while preserving $\text{MEV}_{\mathcal{D}}(S, \mathcal{C})$. Condition (i) is that contract methods 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 (ii) ensures that $\mathcal{D}$ consists enough contracts to reproduce attacks in the stripped state. Condition (iii) 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 $\mathcal{D}\restriction_{\mathcal{C}}$ and $\mathcal{D} \setminus \mathit{deps}(\mathcal{C})$, which could have potentially be exploited by non-wealthy adversaries.

Theorem 4. $\text{MEV}_{\mathcal{D}}(S, \mathcal{C}) = \text{MEV}_{\mathcal{D}\restriction_{\mathcal{C}}}(S, \mathcal{C})$ *holds if the contracts* $\mathcal{C}' = \mathit{deps}(\mathcal{C}) \cap \mathit{deps}(\mathcal{D} \setminus \mathit{deps}(\mathcal{C}))$ *satisfy: (i)* $\mathcal{C}'$ *are sender-agnostic, (ii)* $\mathcal{C}' \subseteq \mathcal{D}$*, and (iii)* $\mathcal{D}\restriction_{\mathcal{C}}$ *and* $\mathcal{D} \setminus \mathit{deps}(\mathcal{C})$ *are token independent in* S*.*

Proof. First note that $\text{MEV}_{\mathcal{D}\restriction_{\mathcal{C}}}(S, \mathcal{C}) \leq \text{MEV}_{\mathcal{D}}(S, \mathcal{C})$ holds by **??**, so we just need to show that

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

To do so, we take a sequence of transactions $\vec{X} \in \kappa_{\mathcal{D}}(\mathcal{M})^*$ that maximizes the loss of $\mathcal{C}$ when executed in state S (we can assume w.l.o.g that $\vec{X}$ is valid in S). We need to show that there is a sequence $\vec{Y} \in \kappa_{\mathcal{D}\restriction_{\mathcal{C}}}(\mathcal{M})^*$ that causes a loss to $\mathcal{C}$ greater or equal to the one caused by $\vec{X}$, i.e. $-\gamma_{\mathcal{C}}(S, \vec{Y}) \geq -\gamma_{\mathcal{C}}(S, \vec{X})$.

To construct $\vec{Y}$, we start by considering $\vec{f}$ to be the sequence of method calls that are performed upon the execution of $\vec{X}$ in state S. Note that $\vec{f}$ includes method calls sent directly from a transaction as well as internal calls that are performed from another called method. We now create a subsequence $\vec{g}$ that contains only the calls that are either:

(a) due to a transaction of $\vec{X}$ directly calling a contract in $\mathit{deps}(\mathcal{C})$, or
(b) due to an internal call in which a method of a contract not in $\mathit{deps}(\mathcal{C})$ calls a method of a contract in $\mathit{deps}(\mathcal{C})$.

Claim (1). A method m whose call appears in $\vec{g}$ due to condition (b) belongs to a contract in $\mathit{deps}(\mathcal{C}) \cap \mathit{deps}(\mathcal{D} \setminus \mathit{deps}(\mathcal{C}))$.

Proof of Claim (1). Clearly m is a method of a contract in $\mathit{deps}(\mathcal{C})$. Moreover, we know that it has been called internally from a method that is not in $\mathit{deps}(\mathcal{C})$, and that this call has originated from a transaction in $\vec{X}$. Such a transaction may only call a contract of $\mathcal{D}$ (since $\vec{X} \in \kappa_{\mathcal{D}}(\mathcal{M})^*$), and we know that it is not calling a method of $\mathit{deps}(\mathcal{C})$ (since $\mathit{deps}(\mathcal{C})$ is closed downward, and m has been called from a method not in $\mathit{deps}(\mathcal{C})$). So, the transaction that originated the call to m must have been targeting a contract in $\mathcal{D} \setminus \mathit{deps}(\mathcal{C})$, meaning that $m \in \mathit{deps}(\mathcal{D} \setminus \mathit{deps}(\mathcal{C}))$. □

We now let $\vec{Y}$ be the sequence of transactions that directly perform the calls in $\vec{g}$, in the same order, with the same arguments and origin. If any of the original calls transferred some tokens, then the corresponding transaction of $\vec{Y}$ will provide the same amount of tokens. Note that this is because the adversary must have enough tokens to fund all the calls in $\vec{Y}$. Indeed, the adversary has enough funds to execute $\vec{X}$, and any token that she gains from the discarded transactions cannot be used by calls in $\vec{Y}$, due to the token independence of $\mathcal{D}\restriction_{\mathcal{C}}$ and $\mathcal{D} \setminus \mathit{deps}(\mathcal{C})$ in S (by assumption).

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

Proof of Claim (2). We have two cases:

1. if a method is in $\vec{g}$ due to (a), then it has been called directly from a transaction in $\vec{X}$, which belongs to $\kappa_{\mathcal{D}\restriction_{\mathcal{C}}}(\mathcal{M})^*$. That transaction can be copied and put into $\vec{Y}$.
2. If a call is in $\vec{g}$ due to (b), then by Claim (1) it belongs to a contract in $\mathit{deps}(\mathcal{C}) \cap \mathit{deps}(\mathcal{D} \setminus \mathit{deps}(\mathcal{C}))$, which is contained in $\mathcal{D}$ by assumption (ii). Moreover, $\mathcal{M}$ is able to craft the arguments of that method by simulating the execution of $\vec{X}$. This means that the transaction $Y \in \vec{Y}$ calling the method belongs to $\kappa_{\mathcal{D}\restriction_{\mathcal{C}}}(\mathcal{M})^*$.

We now need to show that $\vec{Y}$ and $\vec{X}$ modify the state of contracts in Γ in the same way. Note that methods that are in $\vec{g}$ due to (b) are sender-agnostic due to assumption (i) and Claim (1). So, the fact that in the execution of $\vec{Y}$ they are called directly from a transaction, while in the execution of $\vec{X}$ they are called from another contract, does not affect the execution of these methods relatively to $\mathcal{C}$. More in detail, it is important to realize that the sequence $\overrightarrow{h}$ of method calls performed upon the execution of $\vec{Y}$ contains $\vec{g}$ but does not coincide with it, since it also includes all the internal calls that are performed by methods in $\vec{g}$. In fact, $\overrightarrow{h}$ is the subsequence of $\vec{f}$ that contains every call to methods of contracts in $\mathit{deps}(\mathcal{C})$. For this reason, both $\vec{f}$ and $\overrightarrow{h}$ modify the state of contracts $\mathcal{D}\restriction_{\mathcal{C}}$ in the same way; and this implies that $\vec{Y}$ is valid in S and that $-\gamma_{\mathcal{C}}(S, \vec{Y}) = -\gamma_{\mathcal{C}}(S, \vec{X})$ (since contracts in $\mathcal{C}$ are not affected by the absence of contracts outside of $\mathcal{D}\restriction_{\mathcal{C}}$). □

Proof of Theorem 3

Proof. We first note the implicit assumption that $\dagger\Delta$ do not have dependencies in $\dagger\Gamma_{\mathcal{M}}$, i.e. $\mathit{deps}(\Delta) \cap \dagger\Gamma_{\mathcal{M}} = \emptyset$. Since otherwise, the state $S \mid \Delta$ is not well-formed and the thesis does not make sense.

In this proof, we use the notation $\dagger(S \mid \Delta)\restriction_{\dagger\Delta}$ to denote $\mathit{deps}(\Delta)$. Note that this is because $\mathit{deps}(\Delta) \cap \dagger\Gamma_{\mathcal{M}} = \emptyset$, and we can strip away from $S \mid \Gamma_{\mathcal{M}} \mid \Delta$ all contracts in $\Gamma_{\mathcal{M}}$. Furthermore, formally, "the non-dependencies of Δ in state $S \mid \Gamma_{\mathcal{M}} \mid \Delta$" are the contracts $\dagger(S \mid \Gamma_{\mathcal{M}}) \setminus \mathit{deps}(\Delta)$. Hence, Condition (ii) is

equivalent to stating that contracts $\dagger(S \mid \Delta) \restriction_{\dagger\Delta}$ and $\dagger(S \mid \Gamma_{\mathcal{M}}) \setminus \mathit{deps}(\Delta)$ are token independent in $S \mid \Gamma_{\mathcal{M}} \mid \Delta$.

We start by showing the following two equalities, which help us to prove our thesis:

$$\mathrm{MEV}(S \mid \Delta, \dagger\Delta) = \mathrm{MEV}(S \mid \Gamma_{\mathcal{M}} \mid \Delta, \dagger\Delta) \tag{10}$$
$$\mathrm{MEV}_{\dagger\Delta}(S \mid \Delta, \dagger\Delta) = \mathrm{MEV}_{\dagger\Delta}(S \mid \Gamma_{\mathcal{M}} \mid \Delta, \dagger\Delta) \tag{11}$$

Observe that (11) follows directly from Lemma A.1. Indeed, widening the contract state does not increase the MEV extractable from $\dagger\Delta$ since the contracts that $\mathcal{M}$ is allowed to target is fixed. Next, to prove (10), we start by proving the following equality using Theorem 4:

$$\mathrm{MEV}_{\dagger(S \mid \Gamma_{\mathcal{M}} \mid \Delta)}(S \mid \Gamma_{\mathcal{M}} \mid \Delta, \dagger\Delta) = \mathrm{MEV}_{\dagger S \restriction_{\dagger\Delta}}(S \mid \Gamma_{\mathcal{M}} \mid \Delta, \dagger\Delta) \tag{12}$$

By letting $\mathcal{D} = \dagger(S \mid \Gamma_{\mathcal{M}} \mid \Delta)$, we have

$$\mathcal{C}' = \mathit{deps}(\Delta) \cap \mathit{deps}(\dagger(S \mid \Gamma_{\mathcal{M}} \mid \Delta) \setminus \mathit{deps}(\Delta))$$

where $\mathcal{C}' \subseteq \mathcal{D}$ and $\mathcal{C}'$ are sender-agnostic (by assumption). Furthermore, $\dagger(S \mid \Delta) \restriction_{\dagger\Delta}$ and $\dagger(S \mid \Gamma_{\mathcal{M}}) \setminus \mathit{deps}(\Delta)$ are token independent in $S \mid \Gamma_{\mathcal{M}} \mid \Delta$. This implies that all three conditions of Theorem 4 are satisfied and we have proven (12). Next, using Theorem 4 again, we prove the equality

$$\mathrm{MEV}_{\dagger(S \mid \Delta)}(S \mid \Gamma_{\mathcal{M}} \mid \Delta, \dagger\Delta) = \mathrm{MEV}_{\dagger S \restriction_{\dagger\Delta}}(S \mid \Gamma_{\mathcal{M}} \mid \Delta, \dagger\Delta) \tag{13}$$

To prove (13), this time we let $\mathcal{D} = \dagger(S \mid \Delta)$ and we have

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

where $\mathcal{C}' \subseteq \mathcal{D}$ and that contracts in $\mathcal{C}'$ are sender-agnostic (by assumption). Furthermore, the token independence of $\dagger(S \mid \Delta) \restriction_{\dagger\Delta}$ and $\dagger(S \mid \Gamma_{\mathcal{M}}) \setminus \mathit{deps}(\Delta)$ in $S \mid \Gamma_{\mathcal{M}} \mid \Delta$ implies the token independence of $\dagger(S \mid \Delta) \restriction_{\dagger\Delta}$ and $\dagger S \setminus \mathit{deps}(\Delta)$ in $S \mid \Delta$ (to see this, substitute $\Gamma_{\mathcal{M}} = \emptyset$). This means that all three conditions of Theorem 4 are satisfied and we have proven (13). Now we can prove (10) by observing the following chain of equalities:

$$\begin{array}{lll}
\mathrm{MEV}_{\dagger(S \mid \Delta)}(S \mid \Gamma_{\mathcal{M}} \mid \Delta, \dagger\Delta) = \mathrm{MEV}_{\dagger(S \mid \Gamma_{\mathcal{M}} \mid \Delta)}(S \mid \Gamma_{\mathcal{M}} \mid \Delta, \dagger\Delta) & & \text{from (12) and (13)} \\
\quad \| & \quad \| & \text{from Lemma A.1} \\
\mathrm{MEV}_{\dagger(S \mid \Delta)}(S \mid \Delta, \dagger\Delta) & \mathrm{MEV}_{\dagger(S \mid \Gamma_{\mathcal{M}} \mid \Delta)}(S \mid \Gamma_{\mathcal{M}} \mid \Delta, \dagger\Delta) & \\
\quad \| & \quad \| & \text{by Item 4 of Lemma 1 in [6]} \\
\mathrm{MEV}(S \mid \Delta, \dagger\Delta) & \mathrm{MEV}(S \mid \Gamma_{\mathcal{M}} \mid \Delta, \dagger\Delta) &
\end{array}$$

Now, our thesis directly follows from Equation (10) and Equation (11). □