Stefan Kitzler, Friedhelm Victor, Pietro Saggese, Bernhard Haslhofer

Disentangling Decentralized Finance (DeFi) Compositions

Disentangling Decentralized Finance (DeFi) Compositions

Stefan Kitzler
kitzler@csh.ac.at
Complexity Science Hub Vienna
Vienna, Austria

Friedhelm Victor
friedhelm.victor@tu-berlin.de
Technische Universität Berlin
Berlin, Germany

Pietro Saggese
pietro.saggese@ait.ac.at
AIT - Austrian Institute of Technology
Vienna, Austria

Bernhard Haslhofer
bernhard.haslhofer@ait.ac.at
AIT - Austrian Institute of Technology
Vienna, Austria

ABSTRACT

We present the first study on compositions of Decentralized Finance (DeFi) protocols, which aim to disrupt traditional finance and offer financial services on top of the distributed ledgers, such as the Ethereum. Starting from a ground-truth of 23 DeFi protocols and 10,663,881 associated accounts, we study the interactions of DeFi protocols and associated smart contracts from a macroscopic perspective. We find that DEX and lending protocols have a high degree centrality, that interactions among protocols primarily occur in a strongly connected component, and that known community detection cannot disentangle DeFi protocols. Therefore, we propose an algorithm for extracting the building blocks and uncovering the compositions of DeFi protocols. We apply the algorithm and conduct an empirical analysis finding that swaps are the most frequent building blocks and that DeFi aggregation protocols utilize functions of many other DeFi protocols. Overall, our results and methods contribute to a better understanding of a new family of financial products and could play an essential role in assessing systemic risks if DeFi continues to proliferate.

CCS CONCEPTS

• **Applied computing** → **Digital cash**; *Electronic funds transfer*.

KEYWORDS

decentralized finance, blockchain, networks

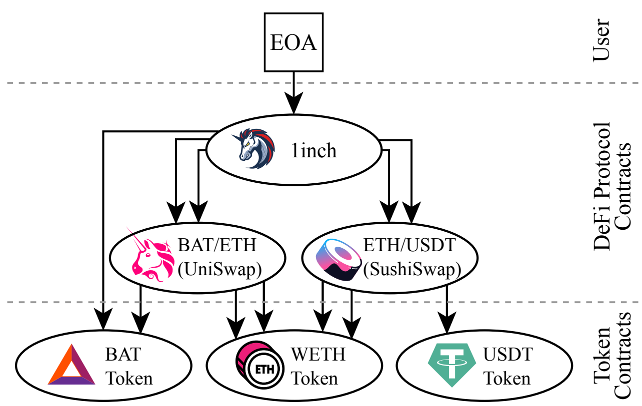

Figure 1: A DeFi composition where *BAT* tokens are swapped against *USDT* tokens through the DeFi service *1inch* in a single transaction. *1inch* executes the swap sequentially through the DeFi services *UniSwap* and *SushiSwap*, using *WETH* as an intermediary token. In the transaction trace graph, we can see the user calling the *1inch* smart contract, which in turn triggers several calls to DeFi protocol-, and token smart contracts.

1 INTRODUCTION

Decentralized Finance (DeFi) stands for a new paradigm that aims to disrupt established financial markets. It offers financial services in the form of *smart contracts*, which are executable software programs deployed on top of distributed ledger technologies (DLT) such as Ethereum. Despite being a relatively recent development, we can already observe rapid growth in DeFi protocols enabling lending of virtual assets, exchanging them for other virtual assets without intermediaries, or betting on future price developments in the form of derivatives like options and futures. The term "financial lego" is sometimes used because DeFi services can be *composed* into new financial products and services.

As an example of DeFi composition, consider Figure 1, which illustrates a user interacting with the *1inch* decentralized exchange (DEX) aggregator Web service[1]. The user holds an amount of *BAT* tokens and wants to swap them to *USDT* tokens. Using the Web application, she creates a transaction against the *1inch* contract, which in turn triggers a sequence of two swaps on two DeFi protocols within the same transaction: from *BAT* to *WETH* on *UniSwap* and thereafter from *WETH* to *USDT* on *SushiSwap*. In this paper, we study such single transaction DeFi interactions and the networks that arise when combining multiple DeFi transactions.

[1] https://app.1inch.io

Motivation. Within the last year, the total value of tokens held by smart contracts underlying the DeFi protocols has reached 96 billion USD [9], a growth rate that central banks increasingly perceive as a risk (cf. [24]). While decentralization of finance offers many opportunities, such as technological innovation or new governance models, it can also undermine established forms of accountability and erode the effectiveness of financial regulation and enforcement. If these protocols are not understood and adopted more broadly, they could have unforeseeable systemic effects on financial markets and our society as a whole, as seen in the 2008 financial crisis [16].

Previous work (cf., [7, 13]) has already shown possible strategies allowing rational agents to maximize their revenues by subverting the intended design of DeFi protocols. However, so far, this has only been discussed within the restricted scope of an individual decentralized exchange or lending protocol. Furthermore, none of the existing studies have systematically investigated compositions of DeFi protocols, which form complex, interconnected financial constructs that can only be understood if we first disentangle them.

Contributions. Our work aims to analyze DeFi protocols and to develop a novel algorithmic method that helps to understand them. We can summarize our contributions as follows:

(1) We provide a manually curated ground-truth of 23 DeFi protocols and 10,663,881 associated smart contracts and construct two network abstractions representing interactions among DeFi protocols and smart contracts (Section 3).
(2) We study intertwined DeFi protocols from a macroscopic perspective by analyzing the topology of both networks. We find that DEX and lending protocols have a high degree centrality and that protocols interactions primarily occur in a strongly connected component. We also find that known community detection algorithms cannot disentangle DeFi protocols, indicating DeFi compositions (Section 4).
(3) We address the microscopic transaction level and propose an algorithm for extracting the building blocks of DeFi protocols. We apply the algorithm to all protocol transactions in our ground-truth, identify the most frequent building blocks, and find swaps being the most frequent ones. We also demonstrate how to disentangle the building blocks of a single protocol using *1inch* as an example (Section 5.1).
(4) We present an overall picture of DeFi compositions by extracting and flattening the entire nested building block structure across multiple DeFi protocols. Then, we apply our algorithm and conduct an empirical analysis showing that DeFi aggregation protocols (*1inch*, *0x* or *Instadapp*) utilize functions of many other DeFi protocols (Section 5.2).

For reproducibility of results, we make our ground-truth dataset, including the labels as well as our source code, openly available at https://github.com/StefanKit/Untangling_DeFi_Composition.

Implications. We believe that our results are an essential contribution towards understanding DeFi compositions. Furthermore, our algorithm can help assess the composition of individual protocols. Considering the volume of the global financial markets, DeFi is still a niche phenomenon. However, if DeFi continues to proliferate and possibly integrate with the traditional financial sector, understanding DeFi compositions will be an important first step in a wider assessment of systemic risks.

2 BACKGROUND AND DEFINITIONS

We now establish preliminary terms and definitions that are used throughout this work and briefly introduce the related works.

2.1 Ethereum Account Types

Ethereum is currently the most important distributed ledger technology (blockchain) for DeFi services [36]. It differs from the Bitcoin blockchain conceptually as it implements the so-called "account model" with two different account types: an **externally owned account** (*EOA*) is a "regular" account controlled by a private key held by some user. A **code account** (*CA*), which is synonymous with the notion "smart contract", is an account controlled by a computer program, which is invoked by issuing a transaction with the code account as the recipient.

A *CA* must always be initially called by an **external transaction** originating from an *EOA*, but a *CA* can itself trigger other *CAs*. In the latter case, the interaction, which is also known as "message", is denoted as an **internal transaction**. Several branches of internal transactions with varying depth can follow an external transaction, resulting in a cascade, which is also called **traces**.

CAs allow users to implement application-layer protocols, which are essentially programs that can follow some standardized interface. **Tokens** are popular CA-based applications and a way to define arbitrary assets that can be transferred between accounts. The program behind a token manages token ownership and can implement a standardized interface like ERC20, which defines functions standardizing token transfer semantics.

2.2 Decentralized Finance (DeFi) Protocol

A **DeFi protocol** is an application-layer program that provides financial service functions such as swapping or lending assets. More technically, we can define it as follows:

Definition 2.1. A DeFi protocol P is a decentralized application that facilitates specific financial service functions defined and implemented by a set of protocol-specific code accounts.

The following properties distinguish DeFi services from traditional financial services: first, they are *non-custodial*, meaning that no intermediary such as a bank or a broker holds custody of a users' funds. Second, they are *permissionless*, meaning that anyone can use existing or implement new services. Third, they are *transparent*, which means that anyone with the necessary technical capabilities and skills can investigate and audit the state of protocols.

2.3 DeFi Protocol Compositions

The fourth, and in this work most crucial property of DeFi protocols is that DeFi protocols are *composable*: *CAs* can call each other, and their individual functions can be arbitrarily composed into new financial products and services ("Financial Lego") [33]. While this analogy is widely used in the literature, to the best of our knowledge, no work investigates *which are* the basic composable building blocks of more complex financial services and how they are related. Harvey et al. [15] refer broadly to composability as asset tokenization and networked liquidity, while Watcher et al. [29] conceive composability narrowly as a repeated wrapping operation of tokens resulting in new derivative products. However, as illustrated before in Figure 1, we note that DeFi compositions also involve *CAs*, which are not tokens. Also, Engel and Herlihy [10] and Tolmach et al. [26] respectively discuss compositions only in the context of automated market makers (AMMs) and of formal verification of *CAs* related to decentralized exchanges and lending services, which is again a very narrow conception. Thus, there is no comprehensive, technically grounded definition for DeFi compositions to the best of our knowledge. For our work, we define it as follows:

Definition 2.2. A DeFi protocol composition occurs when an account leverages one or more accounts belonging to at least another DeFi protocol within a single transaction to provide a novel financial service.

2.4 Related Work

Others studied networks closely related to the ones we investigated before this work: Lee et al. [18] analyzed the local and global properties of interaction networks extracted from the entire Ethereum blockchain statically found heavy-tailed degree distributions. In a follow-up, Zhao et al. [37] analyzed the temporal evolution of Ethereum interaction networks and found that they proliferate and follow the preferential attachment growth model. Furthermore, several studies focus on the network of Ethereum's tokenized assets: Somin et al. [25], for instance, studied the combined graph of all fungible token networks, while Victor and Lüders [28] explored the networks of the top 1,000 ERC20 tokens individually. Fröwis et al. [11] proposed a method for detecting token systems independent of an implementation standard. Also, Chen et al. [5] conducted a systematic investigation of the whole Ethereum ERC20 token ecosystem and analyzed their activeness, purpose, relationship, and role in token trading. However, none of these related works consider networks that represent DeFi Protocols and their relationships.

Another growing body of research concentrates on specific functions offered by individual DeFi protocols or types of protocols. We are aware of many DEX-related measurements focusing on protocol-specific aspects, such as the magnitude of cyclic arbitrage activity [31], the behavior of liquidity providers [32], or the role of oracles as providers of external information [19]. Other studies focus on lending and borrowing services: Perez et al. [21] analyze liquidations and related participants' behavior in the DeFi protocol *Compound*, while Gudgeon et al. [14] compare market efficiency, utilization, and borrowing rates in different lending protocols. Also, Wang et al. [30] provide methods to identify flash loans in three different DeFi providers and measure their related activity. Finally, we are aware that von Wachter et al. [29] investigate composability from an asset perspective and measure composability by identifying the number of derivatives produced from an initial root asset. However, we apply a more technical, service-oriented perspective and consider, to put it simply, a DeFi composition as being a computer program utilizing other programs' functions.

Overall, we are not aware of previous studies providing a comprehensive picture of DeFi compositions across various protocols. We also do not know any work that analyzes in detail the building blocks of individual DeFi protocols. With this work, we want to close this gap.

3 DATASET AND NETWORK CONSTRUCTION

This section describes the data we collected and network abstractions we constructed for subsequent analysis steps.

3.1 Dataset collection

To study DeFi compositions, we are interested in transactions between Ethereum code accounts associated with known DeFi protocols. Thus, we used on-chain transaction data from the Ethereum blockchain and built a ground-truth of known *CAs* and their associations to DeFi protocols.

3.1.1 On-chain transaction data. We used an OpenEthereum client and ethereum-etl[2] to gather all Ethereum transactions from 01-Jan-2021 (block 11,565,019) to 05-Aug-2021 (block 12,964,999), covering the most recent DeFi history. We collected each external transaction and also parsed its cascade of internal transactions, which together gives us the *trace*. For each transaction, we extracted the source and destination account addresses, the transaction hash, the transferred value, the transaction type (call, create, or self-destroy), as well as the trace id, which indexes the transactions by their execution order. Additionally, we collected the method id of the 4-byte input sequence, which allows us to identify the signature of called methods using the 4Byte lookup service[3].

To distinguish between *CAs* and *EOA*, we gathered all code account creation transactions from the first *CA* created on Ethereum until the end of our observation period. We also use these *creation traces* to associate each *CA* with its creator *CA*. In total, we found 46,112,390 *CAs* and used the output byte sequence to identify 324,143 contracts conforming to the ERC20 standard.

3.1.2 Ground-truth data. We focus on the most relevant protocols regarding valuation and gas-burned from 06-Mar-2021 to 05-Aug-2021. We use monthly samples of the top three total-value-locked protocols from DeFi Pulse[4] for each financial service category to define the set of investigated DeFi protocols. We exclude those in the *payment* category because services like Polygon provide off-chain functionality rather than composable financial services or products. Additionally, we consider protocols including *CAs* of the top ten gas burner list[5] in the observation period.

After identifying the most relevant DeFi protocols, we manually collected the *CAs* associated with each protocol. Since this information is not available on the blockchain, we rely on off-chain and publicly available sources like protocol websites and available documentation. We resolved conflicts of duplicated *CA* to protocol assignments and identical names by querying *CA* addresses on Etherscan[6] and uniquely assigned each *CA* address to its original protocol and obtained a unique label. We denote these manually collected data points as *seed data* and make them available as part of our source code repository.

Next, we extended our seed data by implementing a heuristic that uses the creation transactions and identifies the *CAs* deployed by each seed address. By default, all extended addresses inherit the label and protocol assignments from the corresponding seed

[2]https://github.com/blockchain-etl/ethereum-etl
[3]https://www.4byte.directory/
[4]https://defipulse.com/
[5]https://ethgasstation.info/gasguzzlers.php
[6]https://etherscan.io/

Table 1: Ground-truth dataset summary statistics. Seed addresses were collected manually for each DeFi protocol and then heuristically extended.

	Number of addresses	
Protocol type	Seed	Seed extended
Assets	289	1311
Derivatives	390	400
DEX	242	10,397,838
Lending	486	264,262
	1407	10,663,811

address. Combined with our seed data, these extended addresses form our *extended seed data* set. If an extended address conflicts with an existing seed address, we keep the deployed *CA* and remove the seed address. Table 1 summarizes the number of seed and extended addresses collected for each DeFi protocol category. It shows that our automated expansion does not increase the number of addresses associated with DeFi protocols for assets and derivatives. However, it massively expands the dataset for DEXs and lending protocols utilizing automated factory contract deployments. A significant share of the DEX addresses belongs to *1inch* due to the use of gas tokens. For more details on considered DeFi protocols, we refer to Table 6 in the Appendix.

3.1.3 Dataset reduction. As we are only interested in known DeFi protocols, we finally limited and reduced the *traces* data set to the subset *protocol traces*, where the initial external transaction originating from an *EOA* triggers a *CA* address in our extended seed dataset. This reduction allows us to investigate and interpret compositions within the context of known protocols.

3.2 Network construction

In our analysis, we want to understand and discover relations between DeFi protocols and associated *CAs*. For that purpose, as shown in Figure 2, we constructed networks consisting of DeFi traces on two abstraction levels: the lower-level *DeFi Code Account (CA) Network* network and the higher-level *DeFi Protocol Network*.

The DeFi *CA* network includes all known ground-truth *CAs* triggered by external transactions from arbitrary *EOA* addresses and all *CAs* subsequently called by cascades of internal transactions. We note that *CAs* in the network can or cannot be associated with a DeFi protocol in our ground-truth dataset. We construct the network by filtering all internal and external transactions between *CAs* from the *protocol traces*. Since repeated usage of DeFi services results in recurring transaction patterns, we aggregate and count transactions with the same source and destination address.

The DeFi Protocol network represents interactions between protocols. We constructed it by merging all DeFi *CA* vertices associated with the same DeFi protocol into a single node.

We note that we modeled both networks as a directed graph, in which vertices represent either a protocol or a single *CA*. The weighted edges represent the aggregated set of transactions between DeFi protocols or *CAs*.

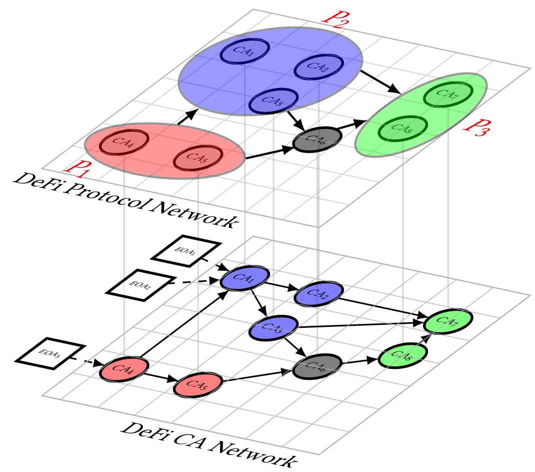

Figure 2: Schematic illustration of constructed networks. The lower-level DeFi Code Account (CA) network represents interactions between *CAs*. The higher-level DeFi Protocol Network models relations between DeFi protocols. Lower-level *CAs* vertices are associated with higher-level protocol vertices. *CAs* are triggered by *EOA* or other *CAs*.

Table 2: Summary statistics of the analyzed networks.

	DeFi CA network	DeFi Protocol network
Nodes	2,536,371	43,624
Edges	3,472,757	84,789
Self-loops	6668	146
Average degree	1.369	1.944
Density	5.398e-07	4.456e-05

4 TOPOLOGY MEASUREMENTS

We now analyze the constructed networks from a macroscopic perspective and investigate whether and how their topological properties are affected by compositions. Table 2 reports basic summary statistics for the DeFi CA network and the DeFi Protocol network. The main difference is in the network dimension, the latter being two orders of magnitude smaller. The presence of self-loops indicates that some contracts include multiple functionalities and thus can also call themselves. Both networks are sparse, as shown by the average degree and density measure, suggesting that *CAs* tend to interact with only a few other *CAs*.

4.1 Degree distribution

Looking at the total-value-locked at DeFi Pulse, we can observe that some DeFi protocols and their contracts play a major role. This observation suggests that they implement core functionality, which other protocols in DeFi compositions can utilize. Under this assumption, preferential attachment [1, 22] is a plausible generative mechanism for both networks. More generally, networks whose degree distribution follows a power law, i.e., the fraction of vertices with degree k is given by $P(k) \sim k^{-\alpha}$ for values of $k \geq k_{min}$, are

Table 3: Likelihood ratio and p-value. None of the reported heavy-tailed distributions is favored over the power law.

	DeFi CA Network	DeFi Protocol Network
Exponential	$\mathcal{R}$: 1.322, p-val: 0.186	$\mathcal{R}$: 4.753, p-val: 0.000
Lognormal	$\mathcal{R}$: -0.406, p-val: 0.685	$\mathcal{R}$: 0.197, p-val: 0.844
Weibull	$\mathcal{R}$: 1.122, p-val: 0.262	$\mathcal{R}$: 2.744, p-val: 0.006

often associated to such generative mechanism. We thus investigate if the power law distribution is a good fit.

We rely on the methodology introduced by Clauset et al. [6] and by Broido et al. [4]. Evidence of scale-free properties exists either when the power law is a plausible model for the distribution or when no alternative heavy-tailed distribution is relatively better than the power law. Thus, we first estimate the parameters $\hat{\theta} = (\hat{k}_{min}, \hat{\alpha})$ by minimizing the Kolmogorov–Smirnov distance between empirical and fitted data for $\hat{k}_{min}$, and exploit it to estimate $\hat{\alpha}$ through the method of maximum likelihood estimation [6]. We then conduct a goodness-of-fit test via a bootstrapping procedure ($N = 5,000$). The resulting p-value indicates if the power law is a plausible fit ($p \geq 0.1$) for the empirical data or not. Finally, we conduct a log-likelihood ratio ($\mathcal{R}$) test to compare the power law fit against other heavy-tailed distributions (i.e., the Exponential, the Lognormal, and the Weibull). A positive value indicates that the power law distribution is favored over the alternative, and the statistical significance is supported by a p-value that indicates if the hypothesis $\mathcal{R} = 0$ is rejected ($p < 0.1$) or not ($p \geq 0.1$).

Figure 3 shows the power law fit for both networks and their estimated $\hat{k}_{min}$ and $\hat{\alpha}$. Coherently with other studies on the interaction networks from Ethereum blockchain data [18], α lies around 1.7 and 1.8, thus being slightly smaller than the average values usually

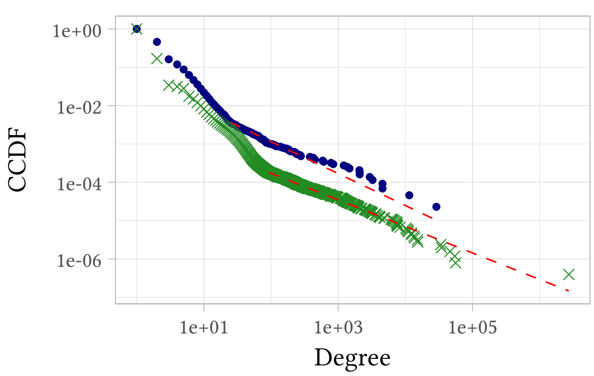

Figure 3: Degree distribution of the CA (×) and Protocol (●) networks. The estimated parameters $\hat{\theta} = (\hat{k}_{min}, \hat{\alpha})$ are respectively $\hat{\theta}_{CA} = (93, 1.69)$ and $\hat{\theta}_P = (25, 1.83)$. In both networks, high-degree nodes are associated to DEX or lending protocols. For the *CA* network, they are routing contracts or factory contracts that deploy other contracts. Nodes with high degree are likely to contain core functionalities and thus to play a relevant role in compositions.

Table 4: Description of the three largest strongly connected components. For both networks the pattern is fragmented, but interestingly the second largest strongly connected components are remarkably more interconnected, indicating that nodes in these components interact with many other nodes, a prerequisite for composition.

		Largest		2nd largest		3rd largest	
	# Comp.	Nodes	Edges	Nodes	Edges	Nodes	Edges
Contract	2,155,707	305,581	611,160	69,116	370,833	5622	11,242
Protocol	33,832	5622	11,242	3948	14,264	36	71

found for power law distributions. The hypothesis that a power law distribution is a good fit is not plausible for both networks because p-values are 0.020 and 0.035 for the CA and protocol networks, respectively. Table 3 reports the comparisons with other heavy-tailed distributions. The power law is not significantly favored over the Lognormal distribution for both networks, while it is a better fit than the Weibull and the Exponential for the protocol network. In summary, according to the classification proposed in Broido et al. [4], both networks have *Super-Weak* scale-free properties.

Furthermore, we found that DEX and lending protocols and their *CAs* have a high degree centrality in both networks. We can hypothesize that they are part of DeFi compositions, which we will explore further in subsequent sections.

4.2 Components

Code accounts of a given protocol could or could not interact with accounts related to other protocols. We thus look at metrics providing further insights on how the (code accounts of) different protocols fall into distinct disconnected components. We distinguish between *weakly* connected components, in which all the nodes are connected by a path independently of the directions of the edges, and *strongly* connected, which considers the edge direction.

For the Protocol network, we find that the largest weakly connected component is equal to the entire network, while for the *CA* network, only 34 nodes are outside of the largest component. The remaining nodes fall into 16 irrelevant components, with a few nodes each. Table 4 lists the three largest strongly connected components. By comparing the number of edges and nodes, we notice that the second-largest component of both the Protocol and the *CA* network is denser than the other larger components. Additionally, in Figure 4 we illustrate how the *CAs* belonging to different protocols map to the ten largest strongly connected components of the *CA* network. Interestingly, the second-largest component also encompasses the vast majority of protocol interactions. While the largest component is entirely composed of *CAs* associated with the *1inch* protocol, in the second-largest component, we find addresses of all the analyzed protocols except for *Renvm*, which is not present in any of the reported large components. We also find that all the protocols fall into the second-largest strongly connected component regarding the Protocol network. This analysis shows that interactions among protocols primarily occur in a single, large component that is more interconnected than average. Notably, such interactions might indicate the existence of compositions.

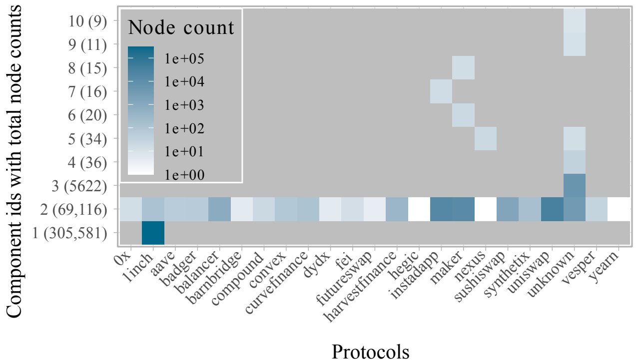

Figure 4: Heatmap showing how the addresses associated to different protocols fall into the ten largest strongly connected components. The largest component is uniquely composed of 305,581 *1inch* addresses, while the second collects the vast majority of protocols. Smaller components identify addresses of protocols that do not interact outside of the protocol itself.

4.3 Community Detection

One could naively assume that *CAs* associated with specific DeFi protocols form communities in the Code Account network. However, the previous results suggest that the network topology reflects DeFi compositions at the level of the community structure. We thus measure how effectively different community detection algorithms detect protocols in the DeFi *CA* network.

We follow the approach of Yang et al. [35], who provide guidelines for selecting community detection algorithms depending on the size of the network. We analyze the weakest largest component in its unweighted and undirected version with non-overlapping communities using four different algorithms: multilevel or Louvain [3], label propagation [23], leading eigenvector [20], and Leiden [27]. Using the labeled addresses in our *ground-truth* dataset, we can verify to what extent $\hat{C}$, the set of communities identified by partitioning algorithms, correspond to P^*, the set of ground-truth communities defined by the individual protocols. We quantify their performance through the *normalized mutual information (NMI)*, a benchmark measure in the literature [8, 17] that quantifies the similarity between the ground-truth communities and the identified communities. In addition, we provide two additional measures: the ratio $\hat{C}/P^*$ for the accuracy of the number of identified communities and the F1 score. We compute the latter similarly to [34]: first, for each protocol $P_i \in P^*$ we identify the detected community $C_j \in \hat{C}$ that maximizes the F1 score. Then, we report average precision, recall, and F1 scores over all communities $P_i \in P^*$. Note that we compute the above metrics only on the labeled *CAs*.

The second column of Table 5 reports the total number of communities that include labeled *CAs*. The NMI is high for all the protocols, indicating that overall the algorithms correctly partition the network: indeed, all algorithms cluster together the *CAs* created by the *1inch deployer* contract, and *1inch* is by far the largest ground-truth community in terms of labeled accounts. On the other hand, the low F1 scores (0.18-0.49) result from a small set of misclassified ground-truth communities (e.g., *Compound*, *DyDx*, *Fei*). Upon closer inspection, we noticed that some protocols map entirely into a few communities dominated by larger protocols (such as *UniSwap* or *Maker*), negatively impacting precision, while others are split into different communities, affecting recall. *1inch* itself has a non-marginal number of addresses that map into other communities.

Table 5: Performance metrics for the community detection algorithms. Low F1 Scores indicate either that the algorithms poorly identify communities, or that the network topology reflects a more complex organisation at the mesoscopic level.

Algorithms	Communities	Precision	Recall	F1 Score	NMI	$\hat{C}/P^*$
Louvain	14	0.3896	0.7181	0.2917	0.9241	0.6087
Leiden	10	0.3021	0.8589	0.2879	0.9620	0.4348
Label prop.	53	0.7107	0.6009	0.4892	0.9404	2.3043
Eigenvector	4	0.1696	0.9070	0.1776	0.9495	0.1739

In summary, we see that algorithms work well, with NMI scores above 0.92. However, when considering the imbalance in our dataset (precision, recall), we find that known community detection algorithms are ineffective in detecting DeFi protocols: the identified community structure reflects a different organization in which protocols are entangled, indicating a composition pattern.

5 MEASURING DEFI COMPOSITIONS

In Section 4, we analyzed the macroscopic network perspective. We now address the microscopic trace level. First, we propose an algorithm to extract possibly nested building blocks of DeFi protocol calls, which may also be used by other DeFi protocols. We then assess the most frequent building blocks our algorithm identifies and illustrate how the DEX aggregator *1inch* uses multiple such building blocks of other protocols. Finally, we flatten the nested structure of building blocks and study the interaction of DEX and lending services.

5.1 Building Block Extraction Algorithm

Building blocks represent possible recurring DeFi service patterns. In order to detect them, we treat transactions as trees of execution traces. We break the trees into subtrees, starting from the tree's leaves, and identify a building block whenever we encounter a node that is part of a protocol. If multiple protocol nodes exist in a tree, the building blocks can be composed of one another. Next, we create a hash of each building block and use those hashes to chain nested tree structures. We aim to identify building blocks that execute the same logic despite being different instances involving different addresses (i.e., a swap with different tokens). We generalize the execution trace trees as follows:

Preprocessing: In contrast to a graph, like in Figure 1, an execution tree can have the same node appearing multiple times as a leaf node, effectively having no cycles. Each edge has a trace id, determining the order of the calls. If a contract address appears in a trace that has been deployed by a factory, we rename it to `$protocol-DEPLOYED`. Furthermore, we rename all contract addresses as `ASSET`, which fulfill the criteria that their smart contract code contains the standard ERC20 token method signatures, and if within the trace, the token contract is called with one such method. This preprocessing assumes that factory deployed contracts and

ERC20 token contracts provide similar functionality. This allows us to generalize the traces, as many similar interactions with various standardized tokens become identical.

Algorithm 1 takes as input a transaction trace tree $G(E, V, t, m)$ with two edge attributes: the trace id t, indicating the order of execution, and m, indicating the method id of the executed call. The second input is a list of seed protocols nodes, such as those described in Section 3.1.2. The algorithm outputs a list of building blocks and hashes of such building blocks. We first setup the output variables in lines 1–2. We then find edges to the protocol nodes in line 3 and extract all further reachable edges of these to obtain edge-induced subtrees in lines 4–5. We filter them in line 6 to include only those with a minimum depth of 2, such that the protocol node has to make further calls. In line 7, we sort the list of subtrees ascendingly based on their depth. This means small trees are at the beginning of the list, and large trees that may contain these smaller trees are at the end. For each subtree (line 8), we compute a hash in lines 9–14, highlighted in gray, akin to a tree kernel. To compute the hash, we first sort the subtree's edges by order of execution in line 10, and then extract the target vertices of each edge in line 11, essentially excluding the original calling node, which could be different in each transaction. For each of those vertices, we compute the outdegree (line 12), and also determine the method id for each edge (line 13). The hash is then computed from the three aforementioned properties in line 14. Using the target vertices, we retrieve the building block from the original tree (line 15), which may contain leaf nodes of building block hashes as replacing subtrees in line 16 can lead to nested building blocks. Finally, we append building block and hash to their lists in lines 17–18. Once all subtrees are processed, the lists are returned in line 20.

Algorithm 1: Building Block Extraction

Inputs : (1) Directed, attributed transaction trace tree $G(V, E, t, m)$ with functions $t : E \rightarrow \mathbb{N}$ assigning a unique trace id, and $m : E \rightarrow \mathbb{N}$ assigning a method id on the edges of the tree,
(2) protocol vertices V_P

Outputs: Lists of building blocks L_B, and hashes L_B^h

1 $L_B \leftarrow ()$; `// Init. list of building blocks`
2 $L_B^h \leftarrow ()$; `// Init. list of building block hashes`
3 $L_e \leftarrow (uv) | \forall uv \in E : v \in V_P$; `// Edges to protocol nodes`
`// For each edge to a protocol, get subtree`
4 $L_E^p \leftarrow (E_i)|$ edges reachable from e_i for each $e_i \in L_e$;
5 $L_G^p \leftarrow (G[E_i]) \mid \forall E_i \in L_E^p$; `// edge induced subtrees`
6 $L_G^p \leftarrow$ filter(L_G^p, by=tree-depth, minimum=2);
7 $L_G^p \leftarrow$ sort(L_G^p, by=tree-depth, how=ascending);
8 **for** $G_S(V_S, E_S, t, m) \in L_G^p$ **do** `// for each subtree`
9 `// Compute building block hash with` V_s', D_s, M_s
10 $E_S' \leftarrow$ sort(E_S, by=$t(E_S)$, how=ascending); `// Sort edges`
11 $V_S' \leftarrow (v_1, ..., v_n) = v | \forall uv \in E_S' : v \in V_S$; `// Vert. list`
12 $D_S \leftarrow deg_{out}(v) | \forall v \in V_S'$; `// Outdegree list`
13 $M_S \leftarrow m(E_S')$; `// Method id list`
14 $h_S \leftarrow$ sha256hash(stringify(V_s', D_s, M_S));
15 $B_S \leftarrow G[V_S']$; `// B. block as vertex induced subtree`
16 replace(what=G_S, in=G, with=h_S);
17 $L_B \leftarrow L_B \parallel B_S$; `// Append building block`
18 $L_B^h \leftarrow L_B^h \parallel h_S$; `// Append building block hash`
19 **end**
20 **return** L_B, L_B^h

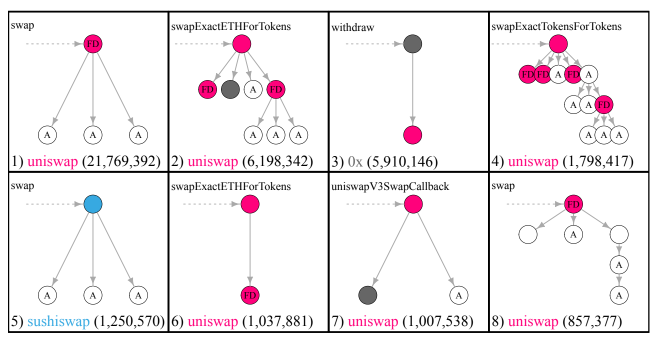

Figure 5: The eight most frequently observed building blocks by called root method, root protocol and count. Nodes marked with FD are generalized factory deployed contracts and those marked with A are ERC20 assets. The majority of these building blocks originate from *UniSwap*. Note that block 1 of *UniSwap* is equivalent to number 5 of *SushiSwap*. This makes sense, as *SushiSwap* is a fork of *UniSwap*. Number 1 is contained in building blocks 2, and 4 – illustrating an internal composition within the same protocol. Building block 3 represents the withdrawal of Wrapped Ether (*WETH*) and is associated to *0x*.

5.2 Building Block Analysis

We execute the algorithm on all transactions in our dataset, together with the set of DeFi protocols in our labeled extended seed set (cf. Section 3). We can then count the retrieved building blocks by their hashes, understand their composition, and visualize them.

Figure 5 illustrates the top 8 most frequently observed building blocks, of which six belong to *UniSwap*. The most frequent building block is a *UniSwap* swap, with more than 21 million occurrences. As *UniSwap* is one of the most popular DeFi protocols, and token swaps are its main functionality, this result shows that the building block extraction is meaningful. ´

5.2.1 Protocol Building Block Composition. Building blocks obtained from Algorithm 1 can contain leaf nodes with hashes that point to other building blocks, leading to a nested structure. We can inspect this structure to observe which protocol building blocks are called from another protocol. For the protocol *1inch*, Figure 6 illustrates which protocol building blocks are called in the first level with a tree map. Each box represents the share of external transactions seen with one or multiple DeFi service building blocks. The colors illustrate the number of unique DeFi services included. In addition to the large *NONE*-share, where no building blocks were detected, *1inch* incorporates a high number of DeFi services from a variety of different protocols. Inspecting the nested building block structure illustrates the composition of multiple DeFi services.

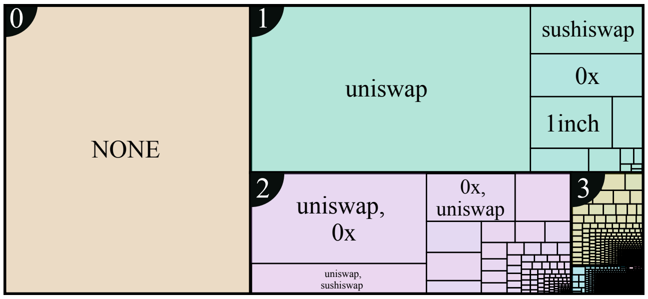

Figure 6: Inspecting the first level of potentially nested building blocks used by *1inch*. The size of each box represents the share external transactions to *1inch* with building blocks of other protocols. In about a third of those transactions, building blocks of one other protocol are used (green box).

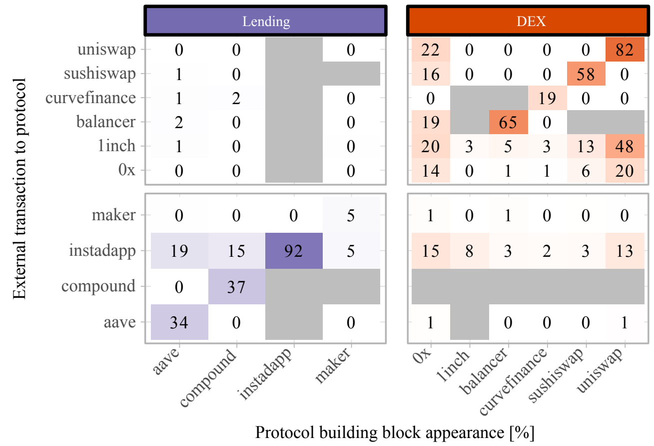

Figure 7: Appearances of DeFi service building blocks across protocols. The numbers indicate the percentage of transactions in which a building block of a certain protocol is contained. The use of multiple DeFi services can be observed for DeFi aggregation protocols, like *Instadapp*, *1inch* and *0x*.

5.2.2 Flattening Composition Hierarchies. Finally, we want to get an overall picture of the DeFi compositions by extracting the entire nested building block structures. In Figure 7 we show the building block appearances of lending and DEX protocols with a heat map. Each line corresponds to an external protocol call, and the line entries indicate the frequency of occurrence of a protocol. The relative share measurement is the fraction of internal building blocks based on the number of external transactions. Most protocol interactions exist within each protocol, visible by the highlighted diagonal elements.

However, DeFi aggregation protocols such as *Instadapp*, *1inch*, and *0x* in particular show extensive use of other DeFi services and thus frequent occurrences of DeFi compositions. This indicates Algorithm 1 works as intended, as, by definition, aggregation protocols must call other protocols. The frequent appearance of the *0x* protocol can be attributed to the popular *Wrapped Ether* token and its *withdraw* pattern, already observed and shown in Figure 5. Finally, we note that second to *0x*, *UniSwap* building blocks appear in most transactions to the protocols shown in Figure 7. The full heat map of all protocols can be found in Figure 9 in the Appendix.

6 CONCLUDING REMARKS

We empirically analyzed 23 DeFi protocols from Jan-2021 to Aug-2021 and constructed network abstractions representing the interactions between smart contracts (*CAs*) and DeFi protocols. We conducted a topology analysis, and the results indicate the existence of compositions. We also found that known community detection algorithms cannot disentangle DeFi protocols. Therefore, we proposed an algorithm that extracts the building blocks of DeFi protocols from transactions. We assessed the most frequent blocks and found that swaps play an essential role. We also analyzed individual DeFi protocols by disentangling their building blocks and flattened the composition hierarchies of all DeFi protocol transactions in our dataset. In summary, our work is the first that investigates DeFi compositions across multiple protocols, both from a network perspective and at the level of individual transactions.

We acknowledge and point out some limitations of our work. First, our results naturally reflect only the compositions of the protocols and labeled addresses contained in our ground-truth dataset. Since the DeFi landscape is evolving rapidly, extending our seed data and the observation period is an obvious next step. One can then re-run our generally applicable analytics procedures. Second, in our network analysis, we currently also neglect edge weights between *CAs*, which may indicate the strength of composition. Including them and investigating temporal evolution could also be part of future work. Third, our building block extraction algorithm currently yields the building blocks of known DeFi protocols. We believe that future work should aim at a more systematic evaluation using a curated ground-truth of DeFi compositions. Finally, we point out that currently, we mainly focus on single-transaction interactions between *CAs*. However, DeFi compositions could also be constructed by *EOA* over time using multiple transactions. We do not yet consider this aspect in our analysis, but we deem it one of the most promising avenues for future work.

Buildings can collapse when individual, load-bearing building blocks fail. This analogy applies to DeFi as well, and we believe that developers and users of DeFi protocols must understand the inner workings of other protocols they are building on. Furthermore, if the DeFi ecosystem evolves at the current pace and integrates closely with the traditional financial sector, associated systemic risks must be understood and mitigated. Thus, we believe that our methods make an essential contribution to understanding the bigger picture and the basic building blocks of individual DeFi protocols and their relationships across protocols. Our work thus also complements the literature on quantifying and managing systemic risks in financial systems (cf. [2, 12]). Up to now, the focus was on risks arising from financial contracts between traditional institutions and risks imposed by single actors to the overall financial system. However, if DeFi continues to proliferate, future actors might be DeFi protocols intertwined with other DeFi protocols. Therefore, we believe that a better understanding of DeFi compositions is an essential first step towards a broader and more systematic study of risks associated with DeFi protocols.

REFERENCES

[1] Albert-László Barabási and Réka Albert. 1999. Emergence of scaling in random networks. *science* 286, 5439 (1999), 509–512.

[2] Marco Bardoscia, Paolo Barucca, Stefano Battiston, Fabio Caccioli, Giulio Cimini, Diego Garlaschelli, Fabio Saracco, Tiziano Squartini, and Guido Caldarelli. 2021. The physics of financial networks. *Nature Reviews Physics* (2021), 1–18.

[3] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment* 2008, 10 (2008), P10008.

[4] Anna D Broido and Aaron Clauset. 2019. Scale-free networks are rare. *Nature communications* 10, 1 (2019), 1–10.

[5] Weili Chen, Tuo Zhang, Zhiguang Chen, Zibin Zheng, and Yutong Lu. 2020. Traveling the Token World: A Graph Analysis of Ethereum ERC20 Token Ecosystem. In *Proceedings of The Web Conference 2020* (New York, NY, USA) *(WWW '20)*. Association for Computing Machinery, 1411–1421. https://doi.org/10.1145/3366423.3380215

[6] Aaron Clauset, Cosma Rohilla Shalizi, and Mark EJ Newman. 2009. Power-law distributions in empirical data. *SIAM review* 51, 4 (2009), 661–703.

[7] Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. 2020. Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 910–927.

[8] Leon Danon, Albert Diaz-Guilera, Jordi Duch, and Alex Arenas. 2005. Comparing community structure identification. *Journal of statistical mechanics: Theory and experiment* 2005, 09 (2005), P09008.

[9] DeFi Pulse. 2021. Total Value Locked (USD) in DeFi. https://defipulse.com/

[10] Daniel Engel and Maurice Herlihy. 2021. Composing Networks of Automated Market Makers. *arXiv preprint arXiv:2106.00083* (2021).

[11] Michael Fröwis, Andreas Fuchs, and Rainer Böhme. 2019. Detecting Token Systems on Ethereum. In *Financial Cryptography and Data Security*, Ian Goldberg and Tyler Moore (Eds.). Springer International Publishing, Cham, 93–112.

[12] Paul Glasserman and H Peyton Young. 2016. Contagion in financial networks. *Journal of Economic Literature* 54, 3 (2016), 779–831.

[13] L. Gudgeon, D. Perez, D. Harz, B. Livshits, and A. Gervais. 2020. The Decentralized Financial Crisis. In *2020 Crypto Valley Conference on Blockchain Technology (CVCBT)*. 1–15. https://doi.org/10.1109/CVCBT50464.2020.00005

[14] Lewis Gudgeon, Sam Werner, Daniel Perez, and William J Knottenbelt. 2020. Defi protocols for loanable funds: Interest rates, liquidity and market efficiency. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*. 92–112.

[15] Campbell R Harvey, Ashwin Ramachandran, and Joey Santoro. 2021. *DeFi and the Future of Finance*. John Wiley & Sons.

[16] Andrei Kirilenko, Albert S Kyle, Mehrdad Samadi, and Tugkan Tuzun. 2017. The flash crash: High-frequency trading in an electronic market. *The Journal of Finance* 72, 3 (2017), 967–998.

[17] Andrea Lancichinetti and Santo Fortunato. 2009. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E* 80, 1 (2009), 016118.

[18] Xi Tong Lee, Arijit Khan, Sourav Sen Gupta, Yu Hann Ong, and Xuan Liu. 2020. Measurements, Analyses, and Insights on the Entire Ethereum Blockchain Network. In *Proceedings of The Web Conference 2020* (New York, NY, USA) *(WWW '20)*. Association for Computing Machinery, 155–166. https://doi.org/10.1145/3366423.3380103

[19] Bowen Liu, Pawel Szalachowski, and Jianying Zhou. 2020. A first look into defi oracles. *arXiv preprint arXiv:2005.04377* (2020).

[20] Mark EJ Newman. 2006. Finding community structure in networks using the eigenvectors of matrices. *Physical review E* 74, 3 (2006), 036104.

[21] Daniel Perez, Sam M Werner, Jiahua Xu, and Benjamin Livshits. 2020. Liquidations: DeFi on a Knife-edge. *arXiv preprint arXiv:2009.13235* (2020).

[22] Derek de Solla Price. 1976. A general theory of bibliometric and other cumulative advantage processes. *Journal of the American society for Information science* 27, 5 (1976), 292–306.

[23] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. 2007. Near linear time algorithm to detect community structures in large-scale networks. *Physical review E* 76, 3 (2007), 036106.

[24] Fabian Schär. 2021. Decentralized Finance: On Blockchain- and Smart Contract-Based Financial Markets. *Federal Reserve Bank of St. Louis Review* 2 (2021), 153–74. https://doi.org/10.20955/r.103.153-74

[25] Shahar Somin, Goren Gordon, and Yaniv Altshuler. 2018. Network Analysis of ERC20 Tokens Trading on Ethereum Blockchain. In *Unifying Themes in Complex Systems IX*, Alfredo J. Morales, Dan Gershenson, Carlosand Braha, Ali A. Minai, and Yaneer Bar-Yam (Eds.). Springer International Publishing, Cham, 439–450.

[26] Palina Tolmach, Yi Li, Shang-Wei Lin, and Yang Liu. 2021. Formal Analysis of Composable DeFi Protocols. *CoRR* abs/2103.00540 (2021). arXiv:2103.00540 https://arxiv.org/abs/2103.00540

[27] Vincent A Traag, Ludo Waltman, and Nees Jan Van Eck. 2019. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific reports* 9, 1 (2019), 1–12.

[28] Friedhelm Victor and Bianca Katharina Lüders. 2019. Measuring Ethereum-Based ERC20 Token Networks. In *Financial Cryptography and Data Security - 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18-22, 2019, Revised Selected Papers (Lecture Notes in Computer Science, Vol. 11598)*, Ian Goldberg and Tyler Moore (Eds.). Springer, 113–129. https://doi.org/10.1007/978-3-030-32101-7_8

[29] Victor von Wachter, Johannes Rude Jensen, and Omri Ross. 2021. Measuring Asset Composability as a Proxy for DeFi Integration. In *International Conference on Financial Cryptography and Data Security*. Springer, 109–114.

[30] Dabao Wang, Siwei Wu, Ziling Lin, Lei Wu, Xingliang Yuan, Yajin Zhou, Haoyu Wang, and Kui Ren. 2021. Towards A First Step to Understand Flash Loan and Its Applications in DeFi Ecosystem. In *Proceedings of the Ninth International Workshop on Security in Blockchain and Cloud Computing*. 23–28.

[31] Ye Wang, Yan Chen, Shuiguang Deng, and Roger Wattenhofer. 2021. Cyclic Arbitrage in Decentralized Exchange Markets. *Available at SSRN 3834535* (2021).

[32] Ye Wang, Lioba Heimbach, and Roger Wattenhofer. 2021. Behavior of Liquidity Providers in Decentralized Exchanges. *arXiv preprint arXiv:2105.13822* (2021).

[33] Sam M. Werner, Daniel Perez, Lewis Gudgeon, Ariah Klages-Mundt, Dominik Harz, and William J. Knottenbelt. 2021. SoK: Decentralized Finance (DeFi). arXiv:2101.08778 [cs.CR]

[34] Jaewon Yang and Jure Leskovec. 2012. Community-affiliation graph model for overlapping network community detection. In *2012 IEEE 12th international conference on data mining*. IEEE, 1170–1175.

[35] Zhao Yang, René Algesheimer, and Claudio J Tessone. 2016. A comparative analysis of community detection algorithms on artificial networks. *Scientific reports* 6, 1 (2016), 1–18.

[36] Dirk A Zetzsche, Douglas W Arner, and Ross P Buckley. 2020. Decentralized finance. *Journal of Financial Regulation* 6, 2 (2020), 172–203.

[37] Lin Zhao, Sourav Sen Gupta, Arijit Khan, and Robby Luo. 2021. Temporal Analysis of the Entire Ethereum Blockchain Network. In *Web Conference 2021 (WWW'21)*.

A SUPPLEMENTAL MATERIAL

We provide here supplemental tables and figures and briefly comment on them.

A.1 Tables on Seed Data and Degrees

In Table 6 we show the number of seed addresses and the extended seed addresses for each protocol and also report the type they belong to. The extended seed heavily increases for some DEXs and lending protocols in comparison with the original seed. In particular, more than 10 million additional *CAs* are associated with *1inch* due to the factory contract that deploys gas tokens. This *CA* also appears in the first position of Table 7, where we report the top 15 *CAs* sorted by highest degree. As one can see, most of the *CAs* are associated with a few DEX and lending protocols (*1inch*, *UniSwap*, *0x*, *Instadapp*, *Maker*), confirming the findings reported in the main body of the paper.

Table 6: Number of seed addresses per protocol before and after the extension procedure.

		Number of addresses	
Protocol type	Protocol	Seed	Seed extended
Assets	badger	64	278
Assets	convex	22	131
Assets	fei	40	37
Assets	harvestfinance	101	803
Assets	renvm	15	15
Assets	vesper	44	44
Assets	yearn	3	3
Derivatives	barnbridge	40	46
Derivatives	dydx	38	38
Derivatives	futureswap	9	10
Derivatives	hegic	8	8
Derivatives	nexus	24	26
Derivatives	synthetix	271	272
DEX	0x	28	50
DEX	1inch	15	10,338,305
DEX	balancer	9	3473
DEX	curvefinance	163	267
DEX	sushiswap	12	1705
DEX	uniswap	15	54,038
Lending	aave	157	166
Lending	compound	67	65
Lending	instadapp	72	32,770
Lending	maker	190	231,261

A.2 Further Insights on Building Block Analysis

Figure 8 shows the number of external transactions, in logarithmic scale, directed to each of our DeFi protocols. The distribution is heterogeneous, and again the most relevant categories are DEX and lending. *UniSwap* is the most frequently appearing one, with a gap of around one order of magnitude to the second one, which is *Maker*.

Finally, Figure 9 provides a complete picture of the heatmap shown in Figure 7, which corresponds to the bottom left corner of the former. While in the main text we focused only on the DEX and lending protocols, where most of the compositions take place, we show also the interactions with the other two types, namely derivatives and assets. The highlighted diagonal elements indicate that protocols mostly contain building blocks involving their own *CAs*. This pattern is especially remarkable for derivative protocols. Consider, e.g., *DyDx*: all external transactions directed to it contain at least one *DyDx* building block. Also, derivatives protocols have little or no further interactions with other protocols, as shown in the row associated with derivatives in the matrix of heat maps. Finally, we notice that the *NONE* category indicates the share of transactions for which no building blocks have been found.

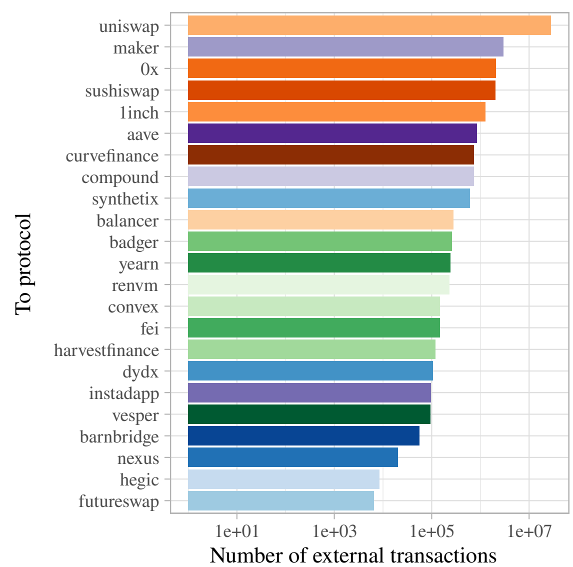

Figure 8: Number of external transactions directed to each protocol. Figures on the x-axis are reported in log scale. *UniSwap* is by far the most prominent one, with more than 10 million calls directed to its *CAs*.

Table 7: First 15 *CAs* by highest degree.

Address	Protocol	Degree	In degree	Out degree
0x0000000000004946c0e9f43f4dee607b0ef1fa1c	1inch	2,713,153	305,627	2,407,526
0x7a250d5630b4cf539739df2c5dacb4c659f2488d	uniswap	56,007	1711	54,296
0xc02aaa39b223fe8d0a0e5c4f27ead9083c756cc2	0x	54,469	45,129	9340
0x5c69bee701ef814a2b6a3edd4b1652cb9cc5aa6f	uniswap	46,408	26,576	19,832
0x2971adfa57b20e5a416ae5a708a8655a9c74f723	instadapp	34,497	18,369	16,128
0x4c8a1beb8a87765788946d6b19c6c6355194abeb	instadapp	33,551	16,956	16,595
0x5ef30b9986345249bc32d8928b7ee64de9435e39	maker	15,300	8940	6360
0x35d1b3f3d7966a1dfe207aa4514c12a259a0492b	maker	15,214	15,214	0
0xa26e15c895efc0616177b7c1e7270a4c7d51c997	maker	13,718	1	13,717
0x0000000000b3f879cb30fe243b4dfee438691c04	unknown	13,447	7644	5803
0x11111112542d85b3ef69ae05771c2dccff4faa26	1inch	12,371	2073	10,298
0x6b175474e89094c44da98b954eedeac495271d0f	maker	12,314	12,314	0
0xdef1c0ded9bec7f1a1670819833240f027b25eff	0x	11,147	1138	10,009
0x939daad09fc4a9b8f8a9352a485dab2df4f4b3f8	instadapp	10,876	10,876	0
0xfd3dfb524b2da40c8a6d703c62be36b5d8540626	unknown	10,554	1547	9007

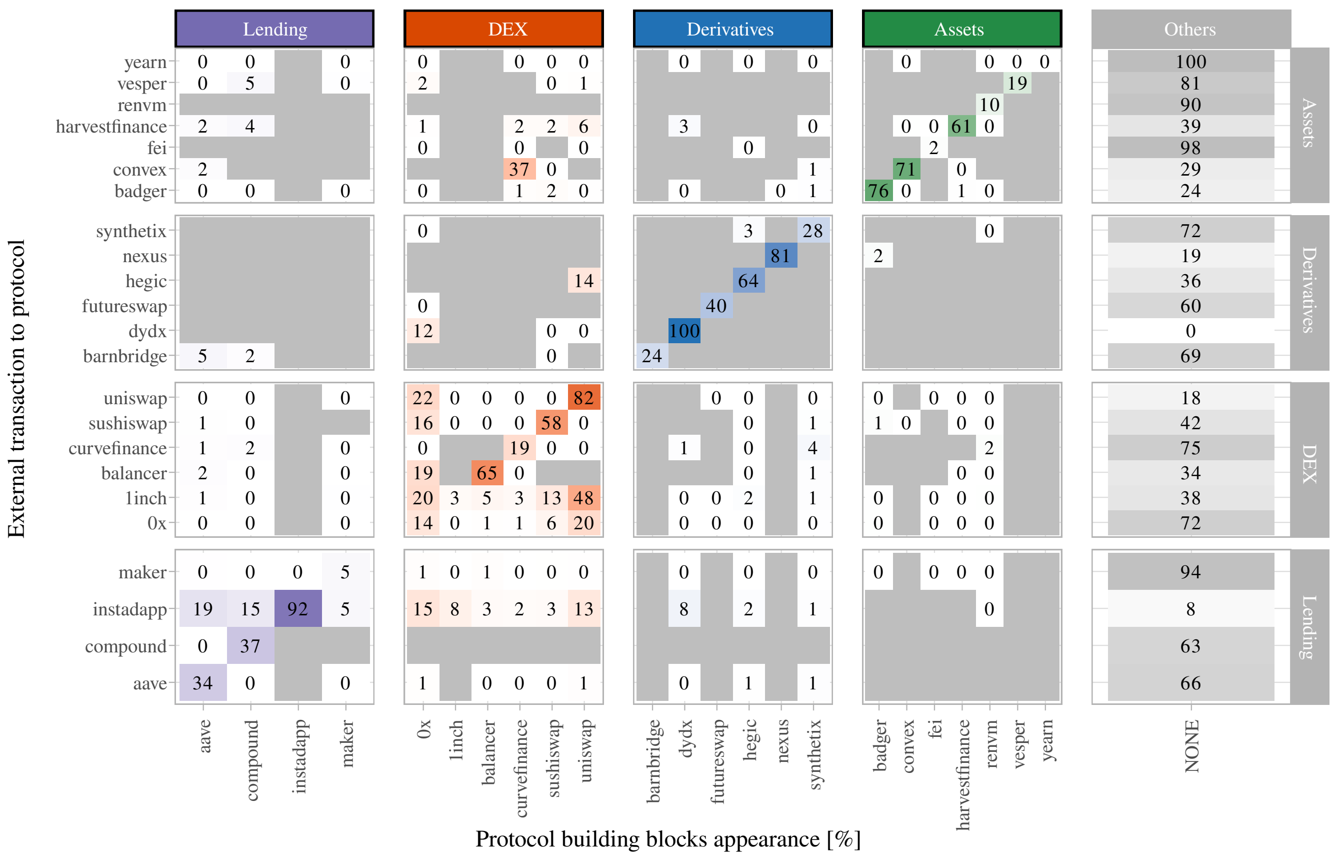

Figure 9: Appearances of DeFi service building blocks across *all* protocols. The numbers indicate the percentage of transactions in which a building block of a certain protocol is observed.