Token Composition: A Graph Based on EVM Logs

Abstract—**Tokens have proliferated across blockchains in terms of number, market capitalisation and utility. Some tokens are tokenised versions of existing tokens — known variously as wrapped tokens, fractional tokens, or shares. The repeated application of this process creates matryoshkian tokens of arbitrary depth. We perform an empirical analysis of token composition on the Ethereum blockchain. We introduce a graph that represents the tokenisation of tokens by other tokens, and we show that the graph contains non-trivial topological structure. We relate properties of the graph, e.g., connected components and cyclic structure, to the tokenisation process. For example, we identify the longest directed path and its corresponding sequence of tokens, and we visualise the connected components relating to a stablecoin and an NFT protocol. Our goal is to explore and visualise what has been wrought with tokens, rather than add yet another brick to the edifice.**

Keywords—**blockchain, tokens, tokenisation, wrapped tokens, decentralised finance, graph analysis**

I. INTRODUCTION

We are witnessing a Cambrian explosion of tokens on blockchains: Ethereum alone has hundreds of thousands of ERC-20 tokens. Many tokens are simple, in the sense that they are not composed of other tokens. But, some are. For example, a liquidity pool token represents a share of a collection of other tokens. DEX Screener [1], a popular liquidity pool tracker, lists over one hundred thousand liquidity pool tokens. Furthermore, tokens are being composed in ever more creative ways: `PT-weETH-25APR2024 (0xb18c87)` is a token issued by Pendle Finance [2] on the Arbitrum blockchain that transitively depends on many other tokens. At the time of writing, this token has a market capitalisation of over one billion US dollars. It is critical from the perspectives of technical and financial risk to examine the composition of tokens. If an investor purchases `PT-weETH-25APR2024`, what tokens does the investment depend upon? In this paper, we present a novel method of examining token composition at both the macro- and micro-level and we apply it to the Ethereum blockchain.

Our method extracts *meta-events* from EVM logs. Low-level events are emitted by contracts. Meta-events are identified by heuristics. A single meta-event can be derived from multiple events. For example, ERC-20 tokens (should) emit a `Transfer` event whenever a token is transferred [3]. A meta-event could signify a token being tokenised by another token, i.e., a deposit of an underlying token with a contract and the minting of a new share, or the burning of a share and the withdrawal of an underlying token from a contract. This meta-event, which we will call a *tokenising meta-event*, can be identified from multiple `Transfer` events within a single transaction. The tokenising meta-events can be represented

as a *token graph*: each vertex represents a token and each directed edge represents the token corresponding to the source vertex being tokenised by the token corresponding to the target vertex. We apply various forms of graph analysis to the token graph and we visualise the structure of token compositions.

This paper is organised as follows. In Sect. II we review related work. In Sect. III we introduce token composition, the token graph, and our data sources. We present our analysis in Sect. IV. Finally, we conclude in Sect. V.

II. RELATED WORK

We categorise related work into four areas: empirical analysis of smart contract composition and code reuse, the automatic detection of tokens, graph analysis of blockchains and token systems, and wrapped tokens.

Software composition is a hard problem [4]. Smart contracts sidestep the low-level problems of interoperability by using a shared execution environment (i.e., a virtual machine) and *de facto* standards (e.g., ERCs), and the high-level problems of architectural mismatch by taking a bottom-up approach to composition. He et al. [5] perform a large-scale analysis of 10 million Ethereum smart contracts deployed between July 2015 and December 2018. They show that less than 1% of the contracts are distinct, and more than 63% of those are similar to at least one other contract. The results have been replicated ([6]–[8]).

Fröwis et al. [9] use symbolic execution to automatically detect smart contracts on the Ethereum blockchain that implement token functionality. Di Angelo and Salzer [10] reconstruct contract interfaces and events from EVM bytecode. They used transaction data to identify token contracts that comply with ERC standards and token contracts that do not. Oliveira et al. [11] propose a taxonomy for classifying tokens and they propose a decision tree to guide the token design process.

Kitzler et al. [12] analyse activity relating to *decentralised finance* (DeFi) on the Ethereum blockchain. They construct and topologically analyse two graphs: the *contract account graph* where the vertices are contract accounts and the edges are transactions between those accounts, and the *protocol graph* where the vertices are protocols and the edges are transactions between those protocols. They show that community finding algorithms identify communities in the contract account graph, but the communities do not correspond to protocols. There are several network analyses of ERC-20 tokens on the Ethereum blockchain that quantify their age, economic value, activity volume, etc. ([13], [14])

Caldarelli [15] describes wrapped tokens and their ability to represent real-world assets and to bridge tokens across blockchains. The WBTC whitepaper [16] sets out a general

framework for tokenising assets on a blockchain. Santoro et al. [17] propose a standard interface for *vaults* for ERC-20 tokens. A vault can store a single *asset* or underlying token. Users can *deposit* or *withdraw* the asset. In return, they receive *shares* in the form of another ERC-20 token.

We use common terminology from graph theory throughout the paper. Please refer to [18] or a similar reference for definitions.

III. TOKEN COMPOSITION

Tokens are central to blockchain-based protocols and applications [19]. Token composition is a technique where one or more tokens can be combined to create new tokens. The entitlements of the input tokens are appropriated by the new token. We extract data from EVM logs (Sect. III-A) and use it to construct a token graph (Sect. III-B), a representation of the tokenisation process. In Sect. IV we will analyse the graph and relate it to the tokenisation process.

A. EVM Logs, Events and Meta-Events

EVM logs record specific occurrences or outputs generated during the execution of contract code. They enable off-chain applications to react to on-chain events. A popular event is the `Transfer` event emitted by ERC 20 tokens [3]:

```
event Transfer(address indexed from,
               address indexed to,
               uint256 value);
```
Listing 1. The ERC 20 `Transfer` event specifies three parameters: `from`, `to`, and `value`.

The event has two special cases. If the `from` address is the zero address (`0x0`) then the contract mints new tokens. If the `to` address is the zero address then the contract burns existing tokens.

A single transaction can emit multiple events. We define a *meta-event* to be a sequence of events that match some pattern and are emitted by a single transaction. We define a *tokenising meta-event* to be a meta-event where the pattern is two `Transfer` events: one must indicate a transfer of tokens to the contract and another must indicate a new token being minted (*deposit & mint*), or one must indicate a token being burned and another must indicate a transfer of an existing token from the contract (*withdraw & burn*). We want to identify instances where a token is tokenised by another. In the terminology of ERC 4626 [17], a tokenising meta-event corresponds to either a `Deposit` event or a `Withdraw` event. However, our tokenising meta-event does not require the contract to follow ERC 4626 and emit those precise events.

We extracted all `Transfer` events from Ethereum mainnet from block height 0 to 16 685 101 (February 2023) inclusive using Geth's `eth_getLogs` RPC method [20]. From the `Transfer` events, we identified 4 032 033 tokenising meta-events using the pattern described above. Table I shows a sample of the data. The first (resp., last) two rows are the earliest (resp., latest) two occurrences of tokenising meta-events in the data that perform a deposit & mint and a withdraw & burn.
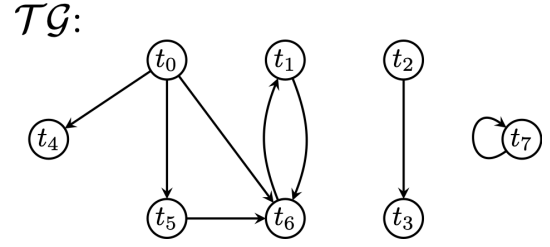

Fig. 1. A token graph, $\mathcal{TG}$, with eight vertices, $t_0, t_1, \ldots, t_7$, and eight edges. Each edge corresponds to a token tokenising another token. For example, the token represented by t_5 tokenises the token represented by t_0.

For example, the first row indicates a transaction that deposited a dust amount of ARC [21] to a contract in a one-to-one exchange for newly minted SWT [22] in January 2017. The third row indicates a transaction that withdrew 5183 BONE [23] from a contract in exchange for burning 5160 tBONE in February 2023. We are not concerned with the individual utility or value of the tokens (or lack thereof); we are only interested in the fact that Token $\mathcal{X}$ was deposited with a contract to mint Token $\mathcal{Y}$, and/or Token $\mathcal{Y}$ was burned by a contract to withdraw Token $\mathcal{X}$.

We filter the tokenising meta-events to include only those meta-events that involve two tokens, Token $\mathcal{X}$ and Token $\mathcal{Y}$, such that there is at least one instance of Token $\mathcal{X}$ being deposited with a contract to mint Token $\mathcal{Y}$, and at least one instance of Token $\mathcal{Y}$ being burned by a contract to withdraw Token $\mathcal{X}$. In other words, the "and/or" conjunction in the previous paragraph is replaced by "and". This excludes *one-way token upgrades* where Token $\mathcal{X}$ can be deposited with a contract to mint Token $\mathcal{Y}$ but Token $\mathcal{X}$ cannot be withdrawn from the contract, and *one-way token burns* where Token $\mathcal{Y}$ can be burned by a contract to withdraw Token $\mathcal{X}$ but Token $\mathcal{X}$ cannot be deposited with the contract to mint Token $\mathcal{Y}$. Of the 4 032 033 tokenising meta-events, 3 461 723 meet the additional requirement. We will refer to the *unfiltered* and *filtered tokenising meta-events* in Sect. III-B.

Additionally, we incorporate market capitalisation data from CoinGecko [24] and liquidity pool data from DEX Screener [1]. CoinGecko aggregates fundamental analysis of tokens including market price, exchange volume, and market capitalisation. DEX Screener [1] stores, parses, and analyses blockchain data to produce a token screener, charts, and analytics.

B. The Token Graph

We can construct a directed graph from the tokenising meta-events as follows. Each vertex corresponds to a token. Each directed edge from a source vertex to a target vertex corresponds to a set of tokenising meta-events that deposits the source token and mints the target token, and/or withdraws the source token and burns the target token. In the terminology of ERC 4626, the source token is the *asset* and the target token is the *share*. Figure 1 shows an example token graph.

Source Token	Target Token	Actions	Source Amount	Target Amount	Tx Hash
ARC (0xac709f)	SWT (0xb12a3c)	deposit & mint	*dust*	*dust*	0x549a12
DGZ (0x84178d)	preDGZ (0x18aa6e)	withdraw & burn	1371	150	0x2da232
BONE (0x981303)	tBONE (0xf7a038)	withdraw & burn	5183	5160	0x5dbe32
WETH (0xc02aaa)	aWETH (0x030ba8)	deposit & mint	25	25	0xb4281a

In the unfiltered case, the directed graph has 23 687 vertices (distinct tokens) and 23 549 edges representing pairs of tokens where the second tokenises the first with either deposit & mint or withdraw & burn actions. In the filtered case, the directed graph has 8424 vertices that are incident with at least one edge and 7536 edges representing pairs of tokens where the second tokenises the first with both deposit & mint and withdraw & burn actions. We will refer to the *unfiltered* and *filtered token graphs* in the remainder of the paper.

C. Data Limitations

Our input data, namely, EVM logs, CoinGecko market data, and DEX Screener liquidity pool data, have limitations. Firstly, EVM logs are unauthenticated: a contract can emit an event of its choosing. There is no guarantee that, say, an ERC-20 `Transfer` event accurately reflects an actual transfer [25]. Furthermore, the first special case highlighted in Sect. III-A (mint) is stipulated by ERC-20[1] but the second (burn) is not. However, EVM logs for the Ethereum blockchain are generally accurate and malicious contracts can be easily excluded. For an aggregated analysis, such as ours, the impact of these limitations should be minimal.

Secondly, the data from CoinGecko and DEX Screener are snapshots that were gathered in April 2024 whereas the EVM logs have a temporal component. It is possible that a token had an entry on CoinGecko in the past, but, at the time the data was gathered, the entry no longer existed. It is also possible that a token was tracked by DEX Screener in the past, but, at the time the data was gathered, it was no longer being tracked. It is also possible that the token coverage of CoinGecko or DEX Screener is incomplete or inaccurate. However, as a high-level measure of token popularity, the impact of the mismatch should be minimal.

Thirdly, tokenising meta-events are a heurstic for identifying instances where one token is tokenised by another. False positives create edges in the token graph where the token corresponding to the source is not tokenised by the token corresponding to the target; false negatives are pairs of tokens where one is tokenised by the other but there is no corresponding edge in the token graph. We will discuss both cases in Sect. IV and Sect. V.

---

[1]"A token contract which creates new tokens SHOULD trigger a Transfer event with the `_from` address set to `0x0` when tokens are created." [3]

Source Token	Target Token	# Tokenising Meta-Events
SHIB (0x95ad61)	xSHIB (0xb4a812)	402 186
BONE (0x981303)	tBONE (0xf7a038)	203 734
SUSHI (0x6b3595)	xSUSHI (0x879824)	120 221
LEASH (0x27c70c)	xLEASH (0xa57d31)	75 180
USDC (0xa0b869)	aUSDC (0xbcca60)	69 373

IV. ANALYSIS

In this section we examine the macro-topological structure of the unfiltered and filtered token graphs including degree distributions, connected component structure, and cyclic structure. We also examine the micro-topological structure of some individual tokens. We visualise the composition of tokens and identify their direct and transitive dependencies.

We note that each edge in a token graph represents a set of tokenising meta-events. Before examining the structure of the graphs, we can identify the most significant edges in terms of the number of tokenising meta-events they contain. Table II shows the results for both the unfiltered and filtered token graphs. Three of the five edges represent the staking of memecoins (SHIB → xSHIB, BONE → tBONE, and LEASH → xLEASH), one represents the staking of a governance token for a decentralised exchange (SUSHI → xSUSHI), and one represents the supply of a stablecoin to a decentralised lending market (USDC → aUSDC). Of course, we could also measure the significance of an edge based on, say, the volume of tokens transacted, the present USD value of the locked tokens, etc.

A. Degree Distributions

The in- and out-degree distributions of the unfiltered and filtered token graphs show an inverse relationship between the degree of a vertex and the number of vertices with that degree (see Fig. 2 and Fig. 3). Table III shows the top five vertices in the unfiltered and filtered token graphs by in-degree and out-degree. The out-degree entries are easy to explain: they are tokens that are deposited with contracts in order to mint many other types of tokens. They include stablecoins (USDC, DAI, USDT and sUSD), wrapped ether (WETH), and wrapped
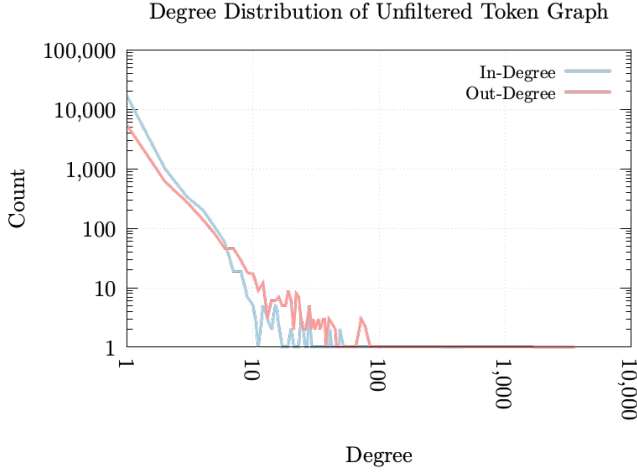
Fig. 2. The in- and out-degree distributions of the unfiltered token graph show an inverse relationship between the degree of a vertex and the number of vertices with that degree. There are a small number of vertices with high degree and a large number of vertices with low degree.
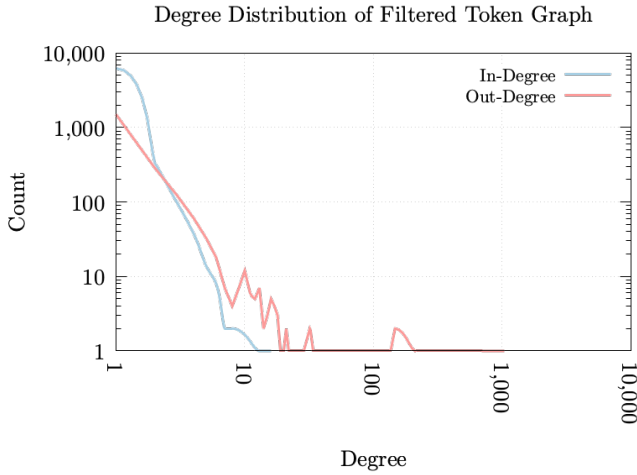

Fig. 3. The in- and out-degree distributions of the filtered token graph show a similar inverse relationship as in Fig. 2.
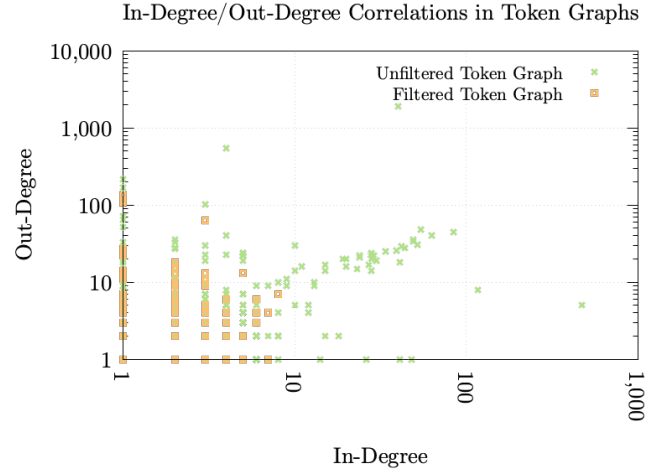

Fig. 4. In the unfiltered and filtered token graphs, we can identify vertices with both high in-degree and high out-degree.

bitcoin (WBTC). The in-degree entries are more complex and have multiple explanations. For example, CHI [26] is a gas token created by 1inch, a decentralised exchange aggregator, that is burned to obtain a reduction in transaction fees; in some transactions the burning of CHI is combined with the withdrawal of another token. This is a false positive generated by our heuristic since CHI does not tokenise a token. The remaining in-degree entries in the unfiltered category are due to token swaps performed during a deposit. For example, aDAI [27] is a yield-bearing token issued by AAVE, a decentralised lending market, in exchange for the stablecoin DAI. However, in some transactions, other tokens are supplied and swapped to DAI. These are also false positives since only DAI is tokenised by aDAI. In the filtered category, the in-degree entries are more reliable. For example, the iDOL token is minted when a user deposits various forms of the SBT token (e.g., SBT09180200, SBT09250200, etc.) according to the Lien Protocol [28]. Similarly, the agEUR token is a stablecoin issued by the Angle Protocol [29] that accepts a variety of tokens as collateral.

Figure 4 plots in-degrees against out-degrees in the unfiltered and filtered token graphs. The tokens whose corresponding vertices have both high in-degree and high out-degree are tokens that tokenise many other tokens, and are themselves tokenised by many other tokens. Examples include mUSD, a stablecoin issued by the mStable protocol [30] and the agEUR token. This makes intuitive sense as stablecoins can be minted from various forms of collateral, and stablecoins can be used as collateral to mint other tokens.

B. Connected Component Structure

The example token graph in Fig. 1 contains three weakly connected components: $\{t_0, t_1, t_4, t_5, t_6\}$, $\{t_2, t_3\}$, and $\{t_7\}$. The unfiltered graph has 4082 weakly connected components. A giant component contains 13 794 vertices (~58%) and 17 711 edges (~75%). There are 3336 components (~82%

TABLE III
THE TOP FIVE VERTICES (TOKENS) IN THE UNFILTERED AND FILTERED TOKEN GRAPHS BY IN-DEGREE AND OUT-DEGREE.

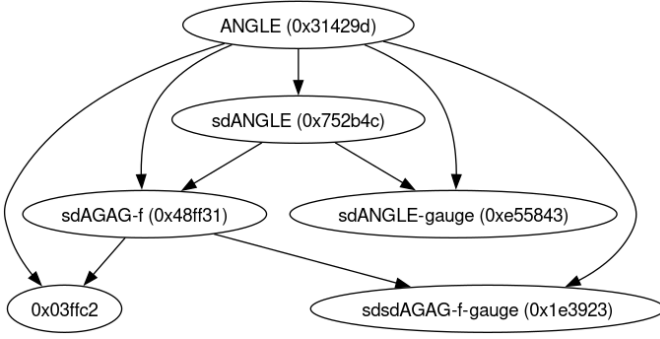|	Top Five by In-Degree		Top Five by Out-Degree	
	Token	Deg.	Token	Deg.
Unfiltered	CHI (0x000000^2)	471	USDC (0xa0b869)	3587
	USDP (0x145668)	117	DAI (0x6b1754)	1923
	aUSDC (0xbcca60)	84	USDT (0xdac17f)	1175
	aWETH (0x030ba8)	63	WETH (0xc02aaa)	951
	aDAI (0x028171)	54	sUSD (0x57ab1e)	548
Filtered	XDP2 (0xe68c1d)	16	USDC (0xa0b869)	1037
	XDP1 (0x134fc6)	15	DAI (0x6b1754)	752
	cyUSD (0x1d0914)	14	USDT (0xdac17f)	396
	iDOL (0x7591a3)	13	WETH (0xc02aaa)	281
	agEUR (0x1a7e4e)	8	WBTC (0x2260fa)	211

Fig. 5. The unfiltered token graph contains a weakly connected component whose vertices correspond to tokens related to the Angle Protocol and Stake DAO.

of the total) with only two connected vertices. These vertices correspond to pairs of tokens where at least one tokenises the other, but neither tokenises, or is tokenised by, a third. None of the tokens represented by the vertices in these components have a non-zero market capitalisation according to CoinGecko and none are traded in any liquidity pool according to DEX Screener. Their lack of popularity reflects their isolation in the token graph.

There are 418 components, other than the giant component, with more than two connected vertices. Figure 5 is an example from this set. It shows the tokenising relationships between tokens related to the Angle Protocol [29] and Stake DAO [31]. Stake DAO implements investment strategies based on other decentralised protocols. Their "Liquid Lockers" generate liquidity, voting power, and yield from lockable tokens. ANGLE is Angle's governance token. It can be deposited with Stake DAO to mint sdANGLE. It is not possible to burn sdANGLE and withdraw ANGLE: the edge between those vertices represents a one-way operation and is not present in the filtered token graph. ANGLE and sdANGLE can be deposited in a gauge (liquidity pool) to mint sdANGLE-gauge. We use this visualisation to identify the direct and transitive dependencies of tokenising tokens such as sdANGLE and sdANGLE-gauge. Many of the other components in the set produce similar insights relating to other tokens and protocols.

The filtered graph has 1491 weakly connected components. A giant component contains 4648 vertices ($\sim$55%) and 5247 edges ($\sim$70%). There are 1162 components ($\sim$78% of the total) with only two connected vertices. There are 196 components, other than the giant component, with more than two connected vertices. Figure 6 is an example from this set that relates to the JPEG'd Protocol [32]. JPEG'd is a decentralised lending market where users can supply non-fungible-tokens (NFTs) as collateral and obtain loans in pETH, a synthetic token that tracks the price of ether. JPEG is the protocol's governance token. The figure shows the various ways in which pETH and JPEG can be tokenised by liquidity pools such as Curve's pETH/WETH pool [33]. Unlike the operations represented by edges in Fig. 5, all operations represented by edges in Fig. 6 can be reversed by either depositing the

underlying and minting the share, or burning the share and withdrawing the underlying.

Both the unfiltered and filtered graphs contain giant weakly connected components. Exploring such structures requires an interactive user interface with navigational aids such as selection, panning, and zooming. As an illustrative example of the interesting structure in the giant components, we present the longest directed path in the filtered token graph. It comprises nine vertices and represents the following sequence of tokens:

1) renBTC (0xeb4c27)
2) sBTC (0xfe18be)
3) crvRenWSBTC (0x075b1b)
4) tbtc/sbtcCrv (0x64eda5)
5) btbtc/sbtcCrv (0xb9d076)
6) ibBTC (0xc4e159)
7) wibBTC (0x8751d4)
8) ibbtc/sbtcCRV-f (0xfbdca6)
9) bibbtc/sbtcCRV-f (0xae96ff)

renBTC can be deposited with a contract to mint sBTC, sBTC can be deposited with a contract to mint crvRenWSBTC, etc. The reverse operations can also be performed (withdraw & burn). The chain involves tokens from several different protocols. It is an example of "composition in the wild" — groups of interacting smart contracts than span multiple protocols.

C. Cyclic Structure

The example token graph in Fig. 1 contains an undirected cycle (t_0, t_5, t_6), a directed cycle (t_1, t_6), and a loop (t_7). Both the unfiltered and filtered token graphs contain undirected cycles. For example, Fig. 5 contains (ANGLE, sdANGLE, sdANGLE-gauge). A priori, it was not apparent if the graphs would contain directed cycles or loops. The filtered token graph contains neither directed cycles nor loops. That is, there is no sequence of n tokens represented by vertices $t_0, t_1, \ldots, t_{n-1}$ such that t_i is tokenised by $t_{i+1 \mod n}$, $0 \leq i < n$. Is this due to a technical limitation in our tokenising meta-event heuristic, or is it the case that no such contract(s) have been deployed during the time period covered by the data? We believe the answer is the latter since it *is* possible to deploy a contract that creates tokenising meta-events that produce a directed cycle in the filtered token graph. We have created such a contract[3] to demonstrate this. It can produce directed cycles of arbitrary length (including loops) in the filtered token graph.

The unfiltered token graph does not contain any loops. However, it does contain a small number of directed cycles. The graph has 50 non-trivial strongly connected components, that is, strongly connected components with more than one vertex. We manually investigated the tokens involved in these directed cycles. Many are "test tokens" or tokens from defunct

---

[3]The contract is published as a GitHub repository. The link is omitted from this submission due to information that can identify the authors. The repository contains a smart contract (ERC20ExchangeWrapper) that can tokenise any token (IERC20 underlyingToken) with any other (IERC20Wrapper overlyingToken).
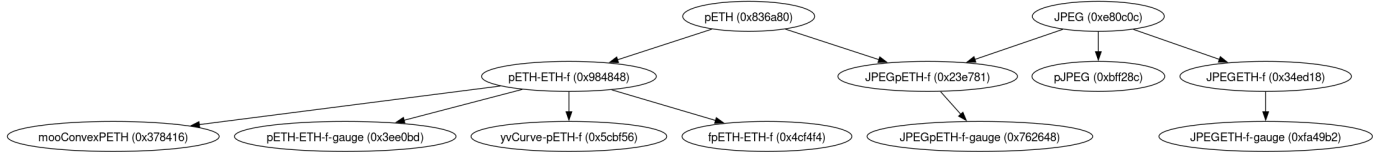
Fig. 6. The filtered token graph contains a weakly connected component whose vertices correspond to tokens related to the JPEG'd Protocol.

protocols, e.g., `TST (0x50e508)` and `TST2 (0x70b34d)`, `TSH (0x46bada)` and `TCH (0x2fe3e4)`, etc. None of the tokens represented by the vertices in these components have a non-zero market capitalisation according to CoinGecko and only one (`USDx (0x2f6081)` and `xBond (0xa8f8dc)`) is traded in a liquidity pool according to DEX Screener. Although directed cycles and loops can occur in a token graph, they are not the product of any practical or widespread operation.

V. CONCLUSION

In many fields, including engineering, chemistry, and cooking, the complexity of a product arises from the combination of numerous base materials or ingredients. This principle holds true for tokens on blockchains. For example, `stkcvxcrvRenWBTC-abra (0xb65ede)` is a token that represents a staked deposit of a share of a liquidity pool for synthetic and wrapped versions of Bitcoin's native token. The base tokens (`renBTC (0xeb4c27)`, `wBTC (0x2260fa)`, etc.) are combined to produce the product. In this paper we detail a novel graph representation of token composition. We construct the graph from the EVM logs of the Ethereum blockchain and we relate its properties to the tokenisation process. For example, we highlight the role of stablecoins that can be minted from various forms of collateral, and can be used as collateral to mint other tokens. In future work, we will refine the heuristic for identifying tokenising meta-events to reduce the number of false positives in the token graph.

REFERENCES

[1] DEX Screener, "DEX Screener," https://dexscreener.com/ and https://docs.dexscreener.com/.
[2] V. Nguyen and L. Vuong, "Standardized yield — A token standard for yield generating mechanisms," https://github.com/pendle-finance/pendle-v2-resources/blob/main/whitepapers/SY.pdf, 2022.
[3] F. Vogelsteller and V. Buterin, "EIP 20: Token standard," https://eips.ethereum.org/EIPS/eip-20, 2015.
[4] D. Garlan, R. Allen, and J. M. Ockerbloom, "Architectural mismatch: Why reuse is still so hard," *IEEE Software*, vol. 26, no. 4, pp. 66–69, 1994.
[5] N. He, L. Wu, H. Wang, Y. Guo, and X. Jiang, "Characterizing code clones in the Ethereum smart contract ecosystem," in *Financial Cryptography and Data Security (FC)*, 2020, pp. 654–675.
[6] M. Kondo, G. A. Oliva, Z. M. J. Jiang, A. E. Hassan, and O. Mizuno, "Code cloning in smart contracts: A case study on verified contracts from the Ethereum blockchain platform," *Empirical Software Engineering*, vol. 25, p. 4617–4675, 2020.
[7] X. Chen, P. Liao, Y. Zhang, Y. Huang, and Z. Zheng, "Understanding code reuse in smart contracts," in *The IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2021, pp. 470–479.
[8] F. Khan, I. David, D. Varro, and S. McIntosh, "Code cloning in smart contracts on the Ethereum platform: An extended replication study," *IEEE Transactions on Software Engineering*, vol. 49, no. 4, pp. 2006–2019, 2022.
[9] M. Fröwis, A. Fuchs, and R. Böhme, "Detecting token systems on Ethereum," in *Financial Cryptography and Data Security (FC)*. Springer, 2019, pp. 93–112.
[10] M. D. Angelo and G. Salzer, "Identification of token contracts on Ethereum: Standard compliance and beyond," *International Journal of Data Science and Analytics*, 2021.
[11] L. Oliveira, L. Zavolokina, I. Bauer, and G. Schwabe, "To token or not to token: Tools for understanding blockchain tokens," in *The International Conference of Information Systems (ICIS)*. ICIS, 2018.
[12] S. Kitzler, F. Victor, P. Saggese, and B. Haslhofer, "Disentangling decentralized finance (DeFi) compositions," https://arxiv.org/abs/2111.11933, 2021.
[13] S. Somin, G. Gordon, and Y. Altshuler, "Network analysis of ERC20 tokens trading on Ethereum blockchain," in *The International Conference on Complex Systems (ICCS)*. Springer, 2018, pp. 439–450.
[14] F. Victor and B. K. Lüders, "Measuring Ethereum-based ERC20 token networks," in *Financial Cryptography and Data Security (FC)*, ser. Lecture Notes in Computer Science (LNCS). Springer, 2019, pp. 113–129.
[15] G. Caldarelli, "Wrapping trust for interoperability: A study of wrapped tokens," https://arxiv.org/abs/2109.06847, 2021.
[16] Kyber Network, BitGo Inc., and Republic Protocol, "Wrapped tokens: A multi-institutional framework for tokenizing any asset," https://wbtc.network/assets/wrapped-tokens-whitepaper.pdf.
[17] J. Santoro, t11s, J. Jadeja, A. C. C. nada, and S. nor Doggo, "EIP 4626: Tokenized vaults," https://eips.ethereum.org/EIPS/eip-4626, 2022.
[18] R. Diestel, *Graph Theory*, ser. Springer Graduate Texts in Mathematics. Springer, 2017.
[19] S. Voshmgir, *Token Economy: How the Web3 Reinvents the Internet*, 2nd ed. Token Kitchen, 2020.
[20] The go-ethereum Community, "go-ethereum (geth)," https://geth.ethereum.org/ and https://geth.ethereum.org/docs.
[21] The Arcade City Community, "Arcade City," https://arcade.city/ https://archive.today/8ayZJ.
[22] The Swarm City Community, "Swarm City," https://swarm.city/ https://archive.today/i2nYE.
[23] The Shiba Inu Community, "Shiba Inu," https://shibatoken.com/ and https://archive.today/AiEPm.
[24] Gecko Labs Pte. Ltd., "CoinGecko," https://coingecko.com/ and https://www.coingecko.com/api/documentation/.
[25] B. Guidi and A. Michienzi, "Sleepminting, the brand new frontier of non fungible tokens fraud," in *The ACM Conference on Information Technology for Social Good*. ACM, 2022.
[26] The 1inch Network, "1inch introduces Chi Gastoken," https://medium.com/1inch-network/1inch-introduces-chi-gastoken-d0bd5bb0f92b.
[27] The Aave Community, "Aave," https://aave.com/ and https://docs.aave.com/.
[28] The Lien Community, "iDOL white paper," https://lien.finance/pdf/iDOLWP_v1.pdf and https://archive.today/RqrSr.
[29] The Angle Community, "Angle," https://angle.money/ and https://docs.angle.money/.
[30] The mStable Community, "mStable," https://mstable.org/ and https://docs.mstable.org/.
[31] The Stake DAO Community, "Stake DAO," https://stakedao.org/ and https://stakedao.gitbook.io/.
[32] The JPEG'd Community, "Jpeg'd," https://jpegd.io/ and https://docs.jpegd.io/jpegd-dao.
[33] The Curve Finance Community, "Curve Finance," https://curve.fi/ and https://curve.readthedocs.io/.