How is Total Value Locked Computed in Decentralized Finance? Towards a Verifiable and Standard Measurement

PIETRO SAGGESE✉, IMT School for Advanced Studies Lucca, Italy
MICHAEL FRÖWIS, Iknaio Cryptoasset Analytics, Austria
STEFAN KITZLER, Complexity Science Hub, Austria
BERNHARD HASLHOFER, Complexity Science Hub, Austria
RAPHAEL AUER, Bank for International Settlements, Switzerland

Total Value Locked (TVL) aims to measure the aggregate value of cryptoassets deposited in Decentralized Finance (DeFi) protocols. Although blockchain data is public, the way TVL is computed is not well understood. In practice, its calculation on major TVL aggregators relies on self-reports from community members and lacks standardization, making it difficult to verify published figures independently. We thus conduct a systematic study on 939 DeFi projects deployed in Ethereum. We study the methodologies used to compute TVL, examine factors hindering verifiability, and ultimately propose standardization attempts in the field. We find that 10.5% of the protocols rely on external servers; 68 methods alternative to standard balance queries exist, although their use decreased over time; 240 equal balance queries are repeated on multiple protocols; and 9.4% of protocols hold tokens that mimic the name or symbol of well-known assets. These findings indicate limits to verifiability and transparency. We thus introduce "verifiable Total Value Locked" (vTVL), a metric measuring the TVL that can be verified relying solely on on-chain data and standard balance queries. A case study on 400 protocols shows that our estimations align with published figures for 46.5% of protocols. To synthesize these dimensions into a single, protocol-level summary, we introduce a composite verifiability score. A majority of protocols already lean toward verifiable computation methods, but about 18.6% obtain a low score. Finally, informed by these findings, we discuss design guidelines that could facilitate a more verifiable, standardized, and explainable TVL computation.

CCS Concepts: • **Applied computing** → **Digital cash**; *Economics*; • **General and reference** → *Measurement*; • **Security and privacy** → *Distributed systems security*.

ACM Reference Format:
Pietro Saggese, Michael Fröwis, Stefan Kitzler, Bernhard Haslhofer, and Raphael Auer. 2018. How is Total Value Locked Computed in Decentralized Finance? Towards a Verifiable and Standard Measurement. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX)*. ACM, New York, NY, USA, 28 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

The core value proposition of Decentralized Finance (DeFi) lies in its transparency and reliance on permissionless on-chain infrastructure [2]. As Total Value Locked (TVL) has become a primary metric for assessing the economic scale of DeFi — measuring the value of assets deposited in smart contracts — it is essential that its calculation upholds the

Authors' Contact Information: Pietro Saggese, IMT School for Advanced Studies Lucca, Lucca, Italy, pietro.saggese@imtlucca.it; Michael Fröwis, Iknaio Cryptoasset Analytics, Vienna, Austria; Stefan Kitzler, Complexity Science Hub, Vienna, Austria; Bernhard Haslhofer, Complexity Science Hub, Vienna, Austria; Raphael Auer, Bank for International Settlements, Basel, Switzerland.

Manuscript submitted to ACM

same principles, remaining fully anchored in on-chain data and fully reproducible calculations. In spite of this, TVL computation today suffers from a lack of standardization and is difficult to verify independently.

In practice, TVL estimates are published by aggregators like DappRadar, Stelareum, and DeFiLlama, which typically adopt a community-driven approach. The latter, for instance, allows anyone to integrate a new DeFi protocol and its TVL computation methodology by developing a protocol-specific plugin and publishing it in an open-source GitHub repository [17]. Contributors are encouraged to use only on-chain data for TVL calculations [18]. However, some plugins rely on data from external services and use self-defined functions for on-chain computations. Moreover, there is variability in how the values of assets deposited in contract accounts are calculated. As of mid 2026, Ethereum TVL estimates published by different aggregators vary from approximately $60 bln to $170 bln, indicating that remarkable differences exist in the methodologies used for TVL computation.

The need for independent verifiability has been demonstrated in cases where DeFi developers on the Solana blockchain deliberately designed their protocols to inflate the actual value of deposited assets, ultimately manipulating TVL figures [40]. The necessity for a standardized approach has become evident with the recognition that crypto deposits can be double-counted across DeFi protocols [7, 9, 41]. This issue has recently been addressed by proposing an alternative metric, Total Value Redeemable (TVR), which refines TVL by excluding cryptoassets that derive their value from underlying cryptoassets [36]. However, a comprehensive understanding of the methodologies used to compute TVL is still lacking [43].

In this paper, we aim to fill this gap by conducting a comprehensive measurement study to examine how TVL is computed in practice and to assess the extent to which its value can be recomputed and verified using on-chain data only. Our contributions and key empirical findings are as follows:

(1) We develop and apply a measurement instrument to 939 DeFi protocols on the Ethereum chain. Our findings reveal that (i) 10.5% of them rely partially or entirely on data from external services to compute TVL, impeding full reproducibility; (ii) while the majority of protocols (78.6%) use standard balance queries, a subset of non-standard, self-defined balance functions ($N = 68$) is employed; (iii) the usage of these alternative queries has declined from 28.2% in January 2023 to 8.9% in January 2024; (iv) 240 balance queries executed on the same contracts and tokens are repeated on multiple protocols; and (v) 9.4% of protocols query contracts that mimic well-known tokens, posing a further threat to transparency.
(2) We introduce *verifiable Total Value Locked (vTVL)*, a metric assessing to what extent individual projects' TVL can be reconstructed using blockchain data and standard balance queries, and the *Discrepancy Ratio*, to quantify differences between published data and our estimates. A case study on 400 protocols shows that discrepancies are negligible for 23.5%, and estimations align with published figures for another 23%. The granularity of vTVL also lets us track the composition of value locked over time, which we find is dominated by wrapped ether and stablecoins, and estimate the economic magnitude of contract-level double counting, which peaked at almost $16 bln in late 2021.
(3) We further synthesize these dimensions into a composite verifiability score, finding that while 52.2% of protocols already lean toward verifiable computation methods, about 18.6% perform poorly.
(4) We propose design guidelines, informed by the challenges we identified, that could lead to a more verifiable and standardized TVL computation: (i) compute TVL from on-chain sources; (ii) publish protocol-specific contracts and tokens lists; (iii) favor standard balance methods whenever possible; (iv) publish the token categorizations used; and (v) define common standards for protocol selection criteria and version management.

As DeFi continues to mature, it is essential to rely on clear and reproducible metrics, especially when these are heavily used for business and investment decisions. Standardization is crucial in this regard, and verifiability should be part of a broader discussion on auditing within the DeFi ecosystem.

Section 2 introduces background and related work; Section 3 discusses how TVL is currently computed, while Section 4 reports the case study. Section 5 introduces the score, while Section 6 and 7 respectively discuss design guidelines and conclusions. Our data and code are available at https://github.com/XXXXXXXXXXXXX/TVL_Study.

2 Background and Related Work

2.1 Cryptoassets: derivative and non-derivative tokens

Cryptoassets represent and facilitate transfer of value in a Distributed Ledger Technology (DLT) [2]. They can be categorized according to different factors [20, 29, 39]. In our context, we distinguish between derivative and non-derivative tokens [36]. Non-derivative or plain tokens are those without an underlying cryptoasset. They include native tokens such as Bitcoin, governance tokens, and non-crypto-backed (NCB) stablecoins, i.e., cryptoassets like USDC whose value is pegged to a target currency and whose reserves are not composed of cryptoassets. Derivative tokens represent instead a receipt token that grants a claim on an underlying cryptoasset. They (non-exhaustively) include liquidity pool (LP) tokens [53], interest-bearing tokens [13, 30], liquid staking tokens (LSTs) [52], and other financial products like synthetic tokens and tokenized baskets of assets. Derivative tokens also include crypto-backed stablecoins such as DAI.

2.2 Total Value Locked

DeFi protocols operate on a peer-to-pool model: investors deposit their cryptoassets into accounts that pool the invested funds to offer financial services [53, 54]. Total Value Locked is the primary metric for assessing the performance of DeFi protocols and the broader DeFi ecosystem, and it is computed as the aggregate value of cryptoassets deposited in the smart contracts that constitute one protocol.

More formally, let $\mathcal{P} = \{p_1, \ldots, p_n\}$ be the set of all DeFi **projects** or **protocols** and $\mathcal{A} = \{a_1, \ldots, a_k\}$ be the set of all **cryptoassets** deployed on a DLT (we remove time subscripts for ease of notation). For one project p, $C^p = \{c_1^p, \ldots, c_m^p\}$ is the set of project-specific contracts. The association of a contract to a project must be *mutually exclusive*, i.e. $C^p \cap C^{p'} = \emptyset$. For each contract $c \in C^p$, a vector $\mathbf{q}$ of length k indicates the amount of tokens locked into the contract, and the price of each token $a \in \mathcal{A}$ is π_a, denominated in a common currency. We then construct the matrices

$$Q^p = \begin{bmatrix} q^p_{c_1,a_1} & \cdots & q^p_{c_1,a_k} \\ q_{c_2,a_1} & \cdots & q^p_{c_2,a_k} \\ \vdots & \ddots & \vdots \\ q_{c^p_m,a_1} & \cdots & q_{c^p_m,a_k} \end{bmatrix}, \qquad \Pi = \begin{bmatrix} \pi_1 \\ \pi_2 \\ \vdots \\ \pi_k \end{bmatrix}, \quad V^p = \begin{bmatrix} v_1^p \\ v_2^p \\ \vdots \\ v_m^p \end{bmatrix}$$

Where $V^p = Q^p \cdot \Pi$. Then TVL^p, denoting the value locked in a protocol, and TVL, denoting the value deposited in a DLT, are respectively computed as:

$$TVL^p = \sum_{i=1}^{m} v_i^p, \qquad TVL = \sum_{p=1}^{n} TVL^p$$

2.3 TVL aggregators

Several aggregators publishing DeFi TVL estimations exist, e.g. DappRadar [14], Stelareum [47], DeFi Pulse [19], Coingecko [11], and DeFiLlama [16]. The estimations they report ultimately rely on on-chain token volumes and price information, but the methods used to compute these figures and the completeness of the related documentation vary greatly.

DeFiLlama is one of the most comprehensive aggregators in terms of TVL information disclosure [36]. Its core infrastructure, the DefiLlama-Adapters GitHub repository [17], is public, so anyone can observe how TVL is computed. New protocols are integrated through individual plugins. While TVL aggregators independently list protocols without any involvement on part of the protocols, and vet every new adapter or pull request before accepting it, the plugins can also be contributed directly by the community — supposedly the protocols' own maintainers — which leads to potential transparency and reliability concerns. The DeFiLlama maintainers discourage the use of data not directly sourced from cryptocurrency nodes (with the exception of exchange rates from Coingecko) [18], but the framework does not impose strict limitations on off-chain data sources. Even when utilizing on-chain data only, the plugin creators can use self-defined, not documented on-chain functions to compute TVL, and the projects do not publish explicitly the contracts and tokens utilized to compute TVL. This information is implicit in the plugin code and no further documentation is provided by protocols. Moreover, the computing code may depend on specific implementations and therefore vary across time. There is also no description informing whether plugin contributors are actual protocol maintainers or other users.

Other aggregators provide more limited documentation, and in some cases, we could not find their TVL-computing code. Some aggregators require DeFi projects to submit a request to be included in TVL calculations, publishing only the final TVL values [19, 47]. These differences can introduce self-selection bias, i.e. the reported TVL may differ consistently across platforms if protocols share information selectively.

Finally, some platforms publish information related to their asset selection criteria, describing what tokens are included in TVL computations [16, 19]. To date, this process is not standardized. Notably, also DeFiLlama's framework allows to select or exclude, e.g., governance tokens, borrowed tokens, and others [9]; its TVL estimations in mid 2026 for Ethereum range from 60 to 170 bln$.

2.4 Related work

Surprisingly, despite the limitations discussed above and the central role TVL plays in DeFi, little research has systematically examined how TVL is computed. This is especially important considering that several works base their analyses on TVL: early overviews of the DeFi ecosystem use TVL as a proxy for the sector's growth [32, 48], while systematic analyses exploit it to quantify the economic scale of losses, e.g. in DeFi security incidents and hacks [55]. A number of econometric studies examine TVL as a driver or correlate of financial outcomes: GARCH models to investigate whether TVL helps explain Ether price dynamics [45]; panel regressions to assess the relationship between TVL and DeFi protocol valuations [38]; clustering and similarity graphs to capture liquidity providers' behavior [27]; logit regressions identifying TVL as negatively related to bubble dynamics in times of market stress [37]. TVL also appears as a key variable to study drivers of DeFi token returns [46]. Scholars tested whether it is positively related to cryptocurrency returns, showing that TVL-based strategies carry no additional explanatory power beyond the aggregate market return [3]. More recent research reveals statistically significant disparities in yield distributions across TVL quantiles, with lower TVL pools consistently offering higher yields [6].

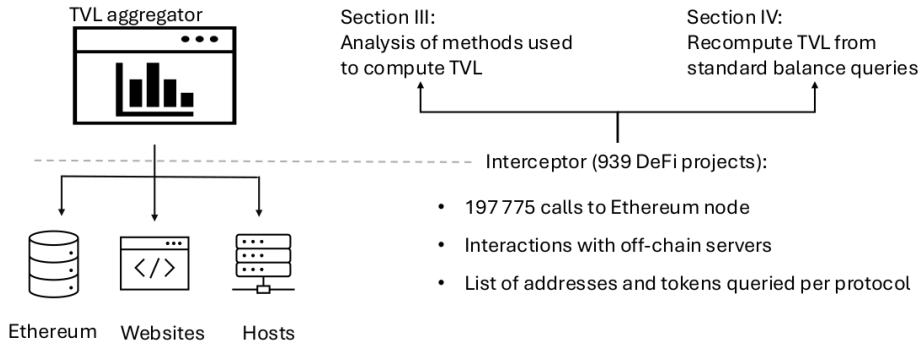


Fig. 1. **Analysis pipeline.** For each protocol, we execute the plugin code provided by the TVL aggregator and record all interactions with both on-chain and off-chain sources.

One major limitation related to TVL computation that was identified by the academic community is the double counting problem [7, 42]. A first solution to this issue was proposed by Luo et al. [36], who focused on asset selection criteria and devised a novel metric to address double counting. *Total Value Redeemable* (TVR) is defined as the value that can be ultimately redeemed from a DeFi ecosystem, i.e., the sum of non-derived tokens deposited in DeFi protocols. By excluding derived tokens from computations, the authors aim to construct a metric that captures more accurately the true underlying value of DeFi. Building on a similar framework, Wu [51] shows that DeFi functions as a layered credit hierarchy mirroring conventional banking, with protocols creating tiers of increasingly derived assets: by late 2025, each dollar of base assets supported $4.7 of total claims. However, rather than treating derivative layering as noise to be filtered out - as TVR approach does - this paper reframes it as structurally informative, showing that token position in the hierarchy impacts yields systematically.

Instead, we are not aware of any prior work assessing comprehensively the reproducibility and verifiability of TVL estimates. Our study fills this gap and complements existing literature [36] by investigating empirically how TVL is effectively computed, what methodologies are used, and the extent to which TVL is reproducible using available on-chain data.

3 TVL Computation in Practice

In this section, we examine how TVL is computed in practice, what data sources are used and what potential challenges to reproducibility and standardization arise from those design decisions. We focus on the projects listed on DeFiLlama for the reasons discussed in Subsection 2.3 (in principle, the analysis could be conducted on other aggregators publishing their TVL-computing code). We restricted the blockchain data collection to Ethereum as it plays a major role in DeFi with 66% of total TVL according to DeFiLlama. We expect that our findings can be generalized to most alternative chains that are EVM compatible and mirror Ethereum's ecosystem. The pipeline of the analysis is shown in Figure 1.

3.1 Measurement method and summary statistics

To conduct our analyses, we developed a data extraction pipeline that proxies and systematically records all interactions between the DefiLlama tooling and its runtime environment during TVL computation.

To integrate a project into DeFiLlama, developers need to create plugins that contain the TVL-computing logic. The DeFiLlama Adapters GitHub repository provides an SDK to simplify this integration process. Once the project

Method	Count	Num. of protocols	Returns the …
balanceOf	102655	641	…account balance of another account (owner address) [23].
eth_getBalance	1696	170	…ETH balance of the account of a given address [24].
totalSupply	2577	154	…total token supply [23].
getReserves	3016	121	…reserves of token0 and token1 used to price trades and distribute liquidity [50].
token0	1040	94	…address of the first pair token [50].
token1	1040	94	…address of the second pair token [50].
symbol	1835	64	…symbol of the token [23].
allPairsLength	46	45	…total number of pairs created through the factory [49].
underlying	1613	43	…address of the underlying token [25].
totalAssets	236	26	…total quantity of all assets under control of a Vault [26].

Table 1. **Most frequently queried functions.** The *balanceOf* method is the most commonly used in TVL computations, along with *eth_getBalance*. Other functions, such as *totalAssets*, offer alternative approaches to retrieving balances.

is successfully integrated, a dedicated folder is created within the repository, containing the protocol plugin and the necessary information to compute TVL. To intercept the interactions, we instrument the runtime environment and execute each project's plugin at the specified Ethereum block height and DeFiLlama commit. [1] In this way, we systematically capture all interactions (via http calls) with the environment involved in generating the TVL for a specific project. This encompasses all calls directed towards the Ethereum node software, as well as interactions with external hosts and web services. Our instrumentation records thus on-chain interactions for arbitrary combinations of Ethereum block height, commit, and chains.

Following the recording process, we filter and interpret the collected on-chain data to identify the functionalities used and accounts accessed per protocol. To interpret the data, we map the signatures of the called functions either directly when stored or using the Ethereum Signature Database 4byte [1].

Our main dataset consists of the interactions between the DefiLlama repository and its environment, collected on January 4 2024 (commit 6764756f9270ab6a3047c06c13c0b1b2d32a3247). It includes 939 projects deployed on Ethereum out of 3494 total projects. It comprises their interactions with external servers and the blockchain infrastructure, the latter resulting in 197775 proxied calls. To validate consistency, we repeated the collection on four other dates (04 Jan 2023, 04 Apr 2023, 04 Jul 2023, 04 Oct 2023).

Table 1 reports the most relevant functions called on-chain, ranked by number of protocols using them and complemented by a short description of their use. As expected, the most frequently called methods are balance queries for ERC-20 compatible tokens (*balanceOf*) and Ether (*eth_getBalance*), but alternative functions exist (e.g., *totalAssets*). *BalanceOf* is primarily queried on wETH, stablecoins, governance tokens, and staked tokens. Other methods are used to query supplementary information, e.g., the token address or its symbol (*token0*, *token1*, *symbol*), are used to price trades and distribute liquidity (*getReserves*), or retrieve token information (*underlying, totalSupply*).

Table 2 reports additional information on the calls directed to the Ethereum node. Specifically, it shows the most common tokens queried in *balanceOf* calls. TVL is computed mostly on wETH and wBTC (respectively 10984 and 1415

[1] Refer to https://github.com/mswjs/interceptors for details on how we intercept all HTTP calls made by the Node.js process.

Address	Symbol	Count	Address	Symbol	Count
0xC02aaA3...	WETH	10984	0x9f8F72a...	MKR	715
0xA0b8699...	USDC	4236	0x6B35950...	SUSHI	690
0xdAC17F9...	USDT	2197	0xae7ab96...	stETH	654
0x6B17547...	DAI	2144	0x7D1AfA7...	MATIC	646
0x2260FAC...	WBTC	1415	0x1111111...	1INCH	616
0x5149107...	LINK	1002	0x408e418...	REN	615
0x1f9840a...	UNI	891	0x4Fabb14...	BUSD	570
0xc00e94C...	COMP	790	0x0F5D2fB...	MANA	564
0x7Fc6650...	AAVE	777	0xc944E90...	GRT	556
0xD533a94...	CRV	760	0x0000000...	TUSD	549

Table 2. **Most frequently called tokens in *balanceOf* functions.** Wrapped ETH and BTC, stablecoins, governance and staked tokens play a primary role.

queries), on stablecoins (USDC, 4236; USDT, 2197; DAI, 2144), governance tokens (UNI, 891; COMP, 790; AAVE, 777), and staked tokens (stETH; 654).

3.2 Reproducibility and reliance on off-chain data

Relying on public and transparent on-chain information entails a number of advantages compared to off-chain data sources. The latter is easier to tamper with, creates more dependencies not in control of the user, and is less transparent, since one has limited visibility into web services' internal operations. We thus run each protocol's computation code and investigate whether the infrastructure relies solely on on-chain data or also exploits external hosts. We also document whether the computation executes correctly or if errors occur.

Figure 2 represents graphically the space of 939 DeFiLlama projects analyzed. For the largest group ($N = 729$), the procedure is executed without errors and no external hosts are used. While 73 projects utilized both external hosts and on-chain data sources, for 26 protocols we observe interactions with external hosts but no direct on-chain data could be retrieved; since no errors occurred in their collection procedure, we infer that they solely rely on external services. The TVL of the remaining protocols is either computed despite errors being produced during computations ($N = 28$) or not computed at all ($N = 62$). Finally, 22 projects did not produce any interaction and were discarded.

Among the 64 external servers identified, some pose a smaller threat to verifiability, e.g., TheGraph, an indexing protocol for accessing blockchain data; others appear to be associated with third parties or specific protocols. The errors are mostly related to technical failures such as the block height provided not being accepted, missing fields or parameters, and asynchronous operations that were not completed successfully. Further details are reported in Appendix A.1.

Arguably, TVL is computable with on-chain data alone, but in practice we find several instances where other data sources are used. Despite the clarity of the DeFiLlama maintainers' objectives, certain protocols (10.5%, including Uniswap V1 and V2) rely partially or entirely on data extracted from external servers for computations, partially compromising verifiability.

3.3 Heterogeneity of on-chain interactions

Even when utilizing on-chain data only, challenges to explainability may arise. Projects use a wide set of functions alternative to standard *eth_getBalance* and *balanceOf* queries to acquire on-chain balance data. These could hinder

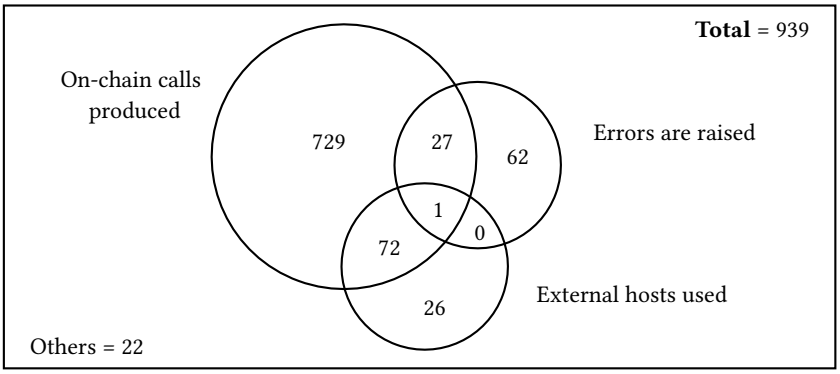


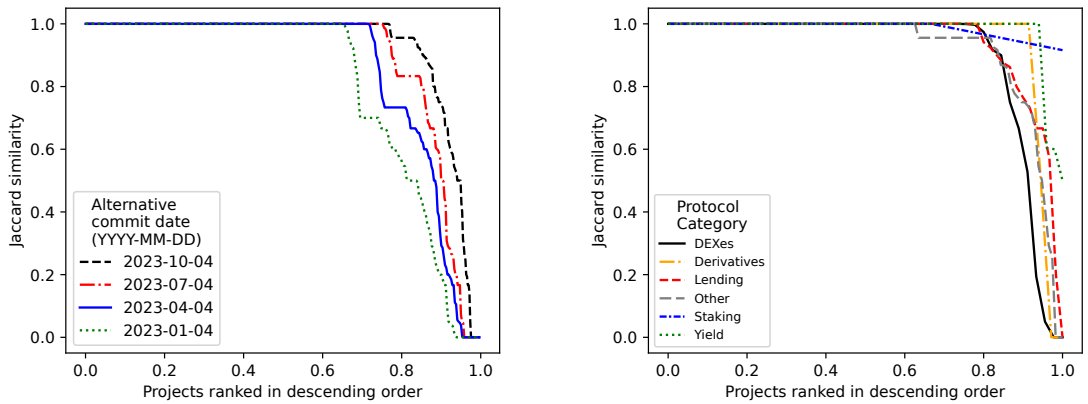
Fig. 2. **Projects using off-chain in addition to on-chain sources.** Projects are categorized based on their reliance on external hosts and whether any errors occurred during data collection. While for 729 projects, on-chain calls were executed without errors and without the use of external sources, 99 projects depend on external servers.

interpretation and introduce sources of tampering or inaccuracies if their logic is unclear. We thus focus on the 197,775 calls directed to an Ethereum node and analyze to what extent the methods used to compute TVL are heterogeneous and the computation approach standardized across protocols.

First, to study if certain functions play an outsized role or if anomalous patterns emerge, we fit to a power-law distribution the frequency of occurrence both of the functions called and of the token calls in *balanceOf* functions. Analyzing the frequency of occurrence of the functions called, we observe that *balanceOf* emerges as an extreme value with respect to the power-law fit, further highlighting its central position in TVL measurement. In general, the results of the fit are consistent with the behavior of a heavy-tailed distribution; the estimated α range from 1.63 to 1.85, coherently with findings of previous network studies on Ethereum blockchain data [34, 35].

Next, we devise an approach to identify the functions alternative to *balanceOf* and *eth_getBalance* (used by 78.6% of protocols) whose name and description indicate that they are likely alternative functions self-defined by projects to compute TVL. We conduct a keyword-based search through regular expressions on the function signature names. We find 68 alternative functions, used by 14.2% of protocols, that plausibly contribute to TVL computation. Out of these, four functions, called 94 times only, have name *balanceOf*, but different signatures than the standard ERC-20. The remaining protocols either rely on external hosts or exploit functions ($N = 88$) that are not clearly matched to a balance-querying functionality after a manual check. While standard functions are simpler to interpret, the alternative functions are specific to a few or just one project, and their logic is hard to interpret without investigating the code in depth. One illustrative example is represented by Lido, whose TVL is computed through one single function (*getTotalPooledEther*) that likely returns as output the total protocol TVL.

In summary, while most protocols (78.6%) use standard token balance queries, a subset of alternative functions ($N = 68$) is employed; however, this makes it harder to understand how TVL is computed for protocols utilizing such functions. Further details on the network analysis and on the detection of the alternative functions are reported in Appendix A.2.



(a) (b)

Fig. 3. **Pairwise Jaccard similarity.** In both plots, similarity is computed by comparing the set of *balanceOf* calls queried in the main and an older commit. Projects are then ranked in descending order by their similarity score (ranging from 0 to 1), where 1 indicates no changes between the compared sets. On the left, values are computed for each protocol on the set of *balanceOf* calls queried in the main and four older commits. Each line reports the values relative to one specific commit. On the right, values are computed for each protocol and broken down by protocol category on the set of *balanceOf* calls queried in the main and the 2023-10-04 commit.

3.4 Changes in TVL computation methods

Another thread to verifiability and consistency is changes to the code itself. TVL computation relies on self-reports from DeFi protocols maintainers, and evidence of manipulations to purposedly inflate TVL exists [40, 41]. More generally, computations may depend on specific implementations and therefore vary across time. As our pipeline enables the data collection for various commits, we repeat the data gathering at four alternative dates (4th of January, April, July, and October 2023) and analyze changes in the Ethereum queries per protocol. To begin, we focus on *balanceOf* calls for their interpretability and measure to what extent projects called different addresses and tokens over time.

To do so, for each protocol we compare the set of *balanceOf* calls executed in the main commit (Jan 4 2024) to the set of *balanceOf* calls executed in each older commit analyzed[2]. We quantify the differences between sets as the pairwise Jaccard similarity; values range from 0 to 1, and a value of 1 indicates no changes among the compared sets. Figure 3a shows the results: each line reports the values relative to one specific commit. Projects are ranked in descending order, based on their Jaccard similarity score. We observe that in each older commit the set of calls does not change for most projects. The average Jaccard index ranges from 0.93 (Oct 2023, black dashed line) to 0.89, 0.86 and 0.81, respectively in Jul, Apr, Jan 2023. The Jaccard similarity decreases as expected with time, but moderately.

These findings indicate that protocols typically do not modify the TVL computation code significantly across time. However, it is worth examining whether this stability is uniform across protocol categories, since the way pools are managed may differ systematically across protocol types: for instance, DEXes may create new trading pairs often and therefore their TVL-computing code may be updated more frequently. Following this intuition, Figure 3b breaks down

[2]We only compare protocols that existed at both commit dates and exclude those that used hosts or raised errors during the data collection. We note that we do not investigate commits executed earlier than 2023 because the number of projects that can be compared decreases with time (from nearly 500 between Oct 2023 and Jan 2024 to less than 300 between Jan 2023 and Jan 2024) and because our pipeline is optimized to capture data on recent implementations of the repository.

the Jaccard similarity scores by protocol category. The results confirm that code stability is a broad phenomenon rather than being confined to a specific segment of the DeFi ecosystem. Derivatives, yield and staking protocols display high similarity scores concentrated near 1, with most projects showing little to no divergence over time, suggesting that the underlying TVL computation logic is rarely revised. DEXes and lending protocols, which together constitute the most relevant categories in DeFi, exhibit a comparable pattern, but slightly more dispersed for high values of the Jaccard index (0.8-1.0), indicating that a higher share of protocols updates frequently their code. On the lower tail, DEXes constitute the larger share of projects updating significantly their TVL computing code (Jaccard index below 0.8), consistent with the hypothesis that DEXes integrate more frequently new trading pairs. The category 'Other' displays a more dispersed distribution, consistent with this category hosting more heterogeneous and experimentally oriented projects whose implementations may evolve more rapidly.

Overall, the cross-category picture corroborates the aggregate finding: the vast majority of DeFi protocols, regardless of their functional category, maintain stable *balanceOf* call sets over the period analyzed. For brevity, Figure 3b reports results for the October 2023 commit; findings are qualitatively consistent across the remaining commits. These results carry a two-sided implication. On the one hand, the stability of TVL computation over time limits concerns about consistency across data collection periods. On the other hand, it may also reflect inertia by protocol maintainers to update their plugins in line with smart contract changes.

Finally, for completeness, we measure how the use of functions alternative to *balanceOf* varies across commits. We find that the ratio between the number of alternative function queries over that of *balanceOf* and *eth_getBalance* calls is higher in older commits (28.2% in Jan 2023, 22.7% in Apr 2023, 19.9% in Jul 2023, and 12.2% in Oct 2023, against 8.9% in Jan 2024). We interpret this as a sign that heterogeneity in computation methods has reduced over time. Appendix A.3 reports additional data, such as the evolution in time of the number of standard and non-standard calls, the full identifiers of older commits, the protocol categorization, and additional results, including the Jaccard similarity scores on all protocols and in absolute numbers.

3.5 Equivalent balance queries linked to multiple protocols

Double counting represents a major issue in the computation of financial metrics like TVL. It occurs when the value of an asset is counted more than once, leading to an inflated representation of total assets within the system. Previous research [36] introduced the Total Value Redeemable as an alternative to TVL and partly addressed the problem by counting the value of non-derivative tokens only. From a technical perspective, double counting can also occur if a smart contract included in the TVL calculation is not exclusively associated with one project, causing the corresponding TVL to be counted multiple times.

We thus search for balance queries that are not exclusively linked to a single protocol, thus potentially leading to such double counting. Notably, we identify 230 *balanceOf* and 10 *eth_getBalance* calls that appear to be associated with different protocols on the same input smart contract and for the same token address. Much of this overlap is concentrated in a restricted number of protocol pairs that appear to be either interconnected or providing bridge solutions. A possible explanation for this finding is that these contracts are managing interactions across such protocols[3]. We measure and discuss the magnitude of this phenomenon in economic terms in the next section. Appendix A.4 reports a full list of the addresses and tokens involved.

[3]While we assume that balances queried during computation directly contribute to the projects TVL, we acknowledge the possibility that they don't.

3.6 Tokens mimicking well-known assets

A further threat to transparency stems from the fact that ERC-20 tokens are easy to replicate. Their names and symbols are arbitrary strings with no guarantee of uniqueness: the univocal identifier of a token is its contract address. This, however, typically remains implicit, while the name and symbol are the human-readable labels that investors primarily use. The concern is twofold. First, it is a quality signal: legitimate protocols overwhelmingly account for value in widely traded contracts (see Table 2), so reliance on homonym tokens points to either misconfiguration or explicit deception. Second, it creates room for value inflation: if the token price retrieved is that of the legitimate original asset, a worthless homonymous token can be valued at the genuine asset's price.

We thus search for queries directed at contracts that mimic well-known tokens. We select the top-200 tokens by query frequency in our dataset, and for each of these reference tokens we identify the contracts that advertise the same name or symbol but are deployed at a different address. We then flag the queries directed at such impersonating contracts, which amount to 541 calls executed by 78 protocols, i.e. 9.4% of those producing on-chain interactions. Out of the top-5 tokens by count in our sample, four of them have homonymous tokens (11 WETH, 8 USDC, 13 USDT, 1 WBTC), with the first three being those with the highest number of homonymous tokens. We note that a subset of these contracts may correspond to forks or test deployments rather than deliberate impersonation; their presence nonetheless complicates verification and warrants scrutiny.

4 Case study: TVL Reconstruction From On-chain Data and vTVL

In principle, TVL can be calculated entirely on public, immutable blockchain information: this may provide a solution to some of the challenges emerged in TVL computation affecting its reproducibility and verifiability. Building on this, we conduct a case study to assess to what extent the TVL of individual projects can be recomputed and verified solely relying on on-chain data and standard balance queries. We call the resulting metric the verifiable Total Value Locked (vTVL) of a DeFi project. This serves as a starting point for discussing a set of recommendations and standardization attempts to improve TVL computation.

4.1 TVL reconstruction approach

Having access to the functionalities and accounts queried on-chain through the DeFiLlama infrastructure, we can acquire a set of addresses per protocol containing deposited assets and provide further insights on the TVL associated with each protocol, solely relying on blockchain data (assuming that addresses called during computation contribute to the project TVL). To have a homogeneous and standardized representation, we focus on *eth_getBalance* and *balanceOf* queries. We obtain a list of addresses contributing towards the locked value for each project, along with a compilation of tokens ($N = 12243$) in which all projects hold value. For each protocol, we extract cryptoassets quantities and prices on-chain. We query the state of protocol-specific addresses for their associated tokens and extract historical monthly balance information from Jan 1^{st} 2021 to Feb 1^{st} 2024. We price tokens by extracting exchange rate information from Uniswap V2 DEX liquidity pools [31]; in total, using this approach, we could price 942 tokens.

Having granular information on tokens deposited into contracts, we can investigate TVL composition and increase control over selected assets. We then categorize tokens following the conceptualization of Section 2 and separate tokens into seven different categories: Ether and its wrapped token wETH, wrapped BTC (wBTC), non-crypto-backed stablecoins, crypto-backed stablecoins, governance tokens, derivative tokens, and others (non-categorized) tokens.

- **Stablecoins**: we extract information from Coingecko [10] and Coinmarketcap [12] on N = 64 stablecoins. We select those with more than 20 mln$ market capitalization, whose price is stable over time, and pegged to the US dollar (stablecoins pegged to EUR or commodities like gold clearly represent a minor market). Next, we analyze their documentation and distinguish 11 decentralized and backed by other cryptoassets[4] from 9 issued by trusted entities and backed primarily by a basket of cash, treasury bills, or other cash equivalents[5].
- **Governance tokens**: we retrieve 110 governance tokens from Coingecko and Coinmarketcap; we complement the list with a regular-expression-based search on our token dataset and label additional 52 tokens that contain the word 'governance' in their name;
- **Bitcoin and Ether wrappers**: we treat separately WBTC and WETH given their particular role in the system. WBTC is by far the largest Bitcoin wrapper compared to other wrappers, which have market capitalization orders of magnitude smaller than BTC. WETH enables ETH to be used in the DeFi ecosystem as an ERC-20 token, and thus we categorize it together with ETH itself in the following.
- **Derivative tokens**: we conduct regular expression searches on token names. Most derivative tokens follow specific patterns; for instance, DEX LP token names typically contain terms like 'LP' or 'Pool'; interest-bearing tokens from protocols like Compound and Aave produce cTokens and aTokens that represent a claim on the underlying asset; similarly, sTokens are receipt tokens for the Synthetix protocol. We further validate this list by adapting the tier-based method proposed by Wu [51], which builds a token derivation hierarchy where base assets occupy the lowest tier and each step of derivation adds a level, and label additional 85 tokens, for a total of 2308 derivative tokens.

We thus recompute the value held by each protocol and call these estimates the verifiable Total Value Locked (vTVL). For comparison, we also download historical TVL values[6] published per protocol from the DeFiLlama API service [15]. We compute for each project the *Discrepancy Ratio*, defined as the average ratio of the vTVL estimations over the data posted through the DeFiLlama APIs, normalized by subtracting one. A value of zero indicates perfect correspondence, while -1 indicates that vTVL equals zero. To interpret this metric, we recall that our estimations are an underestimation rather than an overestimation of the reported figures. We stress that the discrepancies should not be interpreted as a measure that protocols are inflating values; rather, they indicate to what extent we are able to independently verify the reported figures. We then report separately the balance queries that, as discussed in Subsection 3.5, are non-exclusively associated with one protocol and therefore potentially contribute to double counting, as we could not disentangle which project they should be associated with. Finally, we leverage the granularity of our data to analyze the composition of value locked in DeFi and produce a dominance metric that ranks queried tokens by usage over time. Further details on the procedure are given in Appendix A.5.

4.2 Case study: analysis and results

We analyze 400 protocols with at least one *eth_getBalance* or *balanceOf* recorded call and for which we could gather off-chain API data. Panels (a) to (f) of Figure 4 show the results for six protocols selected for their relevance in the DeFi ecosystem: in order, Aave v1, Aave (v2 and v3), Compound-v3, dYdX, Maker, and Uniswap-v3. Each panel shows a stacked plot of the vTVL divided by token categories as discussed above. The black line represents the (total) TVL value

[4]DAI, CRV USD, Liquity USD, FRAX, Ethena USDe, Decentralized USD, mkUSD, sUSD, Alchemix USD, DOLA, MIM.
[5]USDT, USDC, True USD, BUSD, PAX Dollar, First Digital USD, Paypal USD, Gemini Dollar, Verified USD.
[6]The API data are reported by DeFiLlama distinguished by chain and type. We include all values associated with the Ethereum blockchain (columns 'Ethereum', 'Ethereum-borrowed', 'Ethereum-pool2', 'Ethereum-staking', 'Ethereum-vesting').

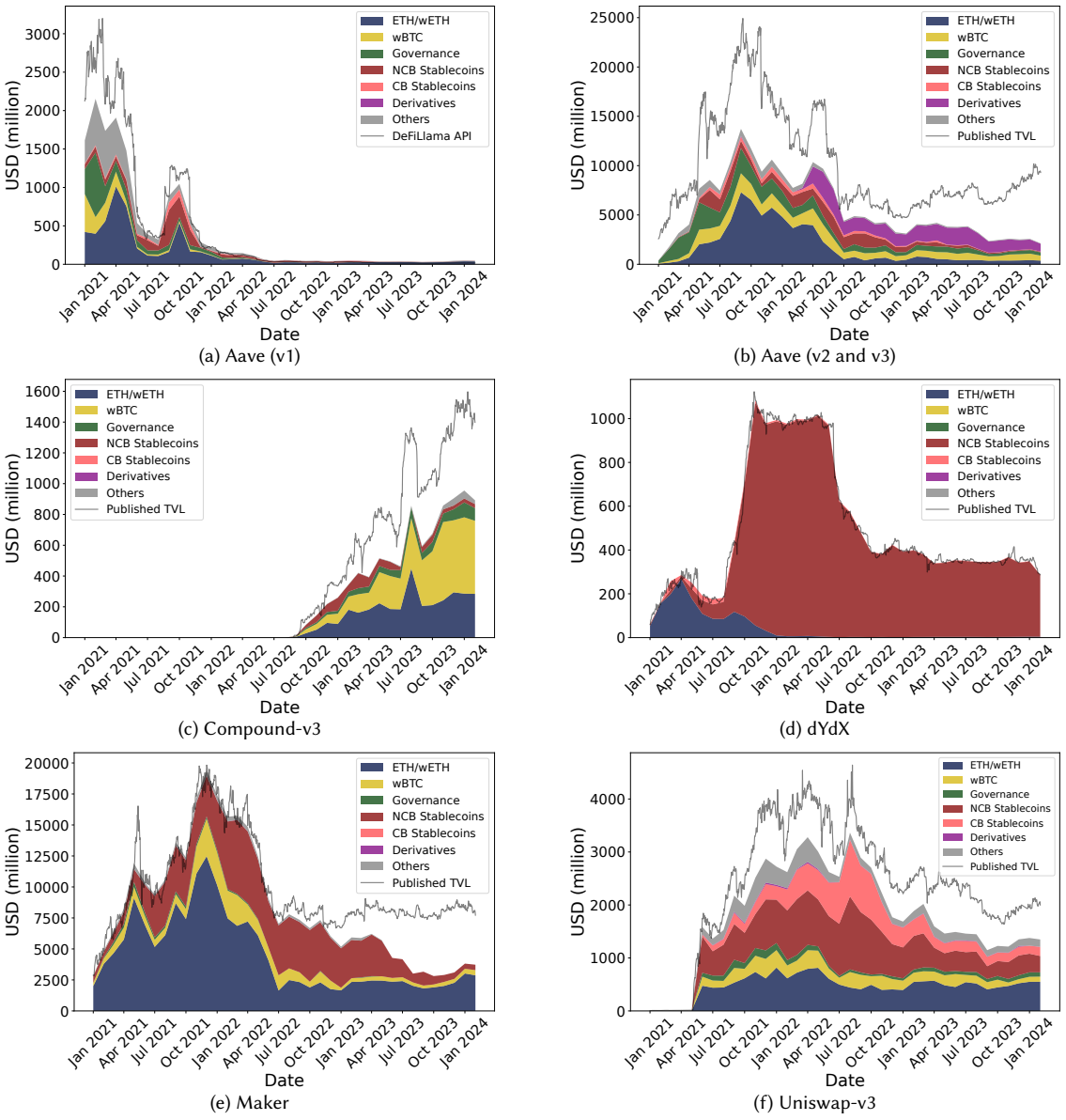


Fig. 4. **Verifiable Total Value Locked (vTVL).** We query on-chain information for relevant DeFi protocols — Aave v1 (a), Aave v2 and v3 (b), Compound v3 (c), dYdX (d), Maker (e), Uniswap v3 (f) — and compare it to published off-chain TVL data. Each plot represents the evolution in time of their vTVL as a stacked plot, divided in seven categories: Ether and wETH, wBTC, governance tokens, non-crypto-backed stablecoins, crypto-backed stablecoins, derivative tokens, and uncategorized tokens.

published on DeFiLlama. We observe that vTVL is mostly composed of ETH/wETH, wBTC, and non-crypto-backed stablecoins. In most cases, the data reconstructed from on-chain and off-chain data are consistent; for some projects, the data match almost perfectly, while for others, we observe a partial discrepancy, but their overall trend is consistent. The

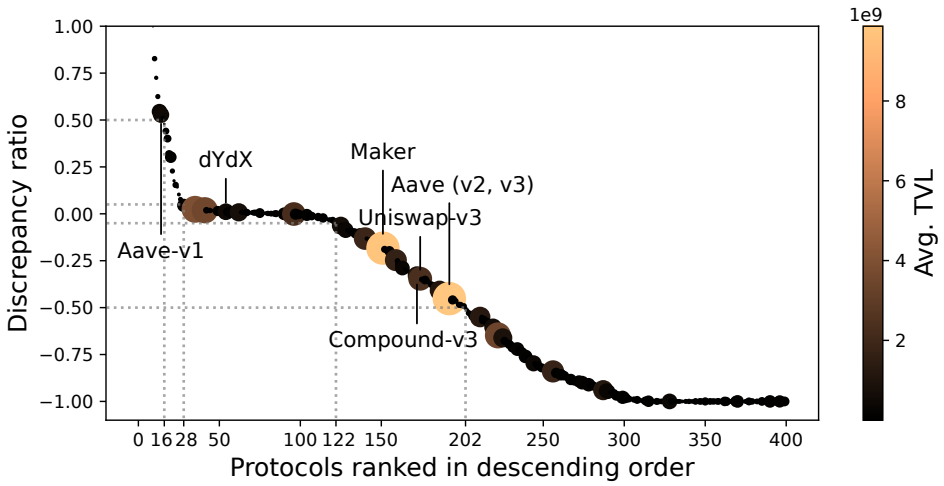


Fig. 5. **Differences between vTVL and published TVL**. Projects are dots ranked in descending order based on the *Discrepancy ratio*, i.e., the ratio of on-chain estimations to off-chain data, normalized by subtracting one.

discrepancies can arise for varying reasons, e.g., the use of alternative methods to compute balances, the reliance on external hosts to produce values (e.g. Maker) or the presence of errors during the interception procedure (e.g. Uniswap V3), but also for the lack of price data for certain tokens. Further details are reported in Appendix A.5 and limitations are discussed in Section 6.

Finally, to obtain a broader understanding of the entire ecosystem, we compute for each project the *Discrepancy Ratio*. Figure 5 shows the results. Each dot represents a project and its size is proportional to the amount of TVL they hold (according to off-chain estimations). Projects are ranked in descending order with respect to the Discrepancy ratio. The ones shown in Figure 4 are labeled. For 94 projects (23.5%), the difference lies within ±0.05, indicating almost perfect consistency, while for 186 projects (46.5%), the difference lies within ±0.5, indicating that figures are aligned but discrepancies exist. In 98 cases the ratio is either larger than 1 (top left) or equal to −1 (bottom right), indicating large discrepancies. The latter are mostly small and less relevant protocols.

In summary, the case study shows that with vTVL, which reproduces TVL using solely on-chain data and interpretable balance queries, we can independently verify just a part of the reported TVL. Discrepancies between our estimations and published data are close to zero for about one quarter of protocols (23.5%) and are aligned for about half (46.5%). Concurrently, the study shows that it is possible to follow a more transparent approach for computing TVL. The approach we used for vTVL removes potential off-chain sources of tampering, minimizes heterogeneity in computation methods, reveals patterns that can only be investigated on-chain, like the association of one contract to multiple protocols, and enables control over the selected assets, ultimately improving reproducibility, verifiability, transparency and interpretability.

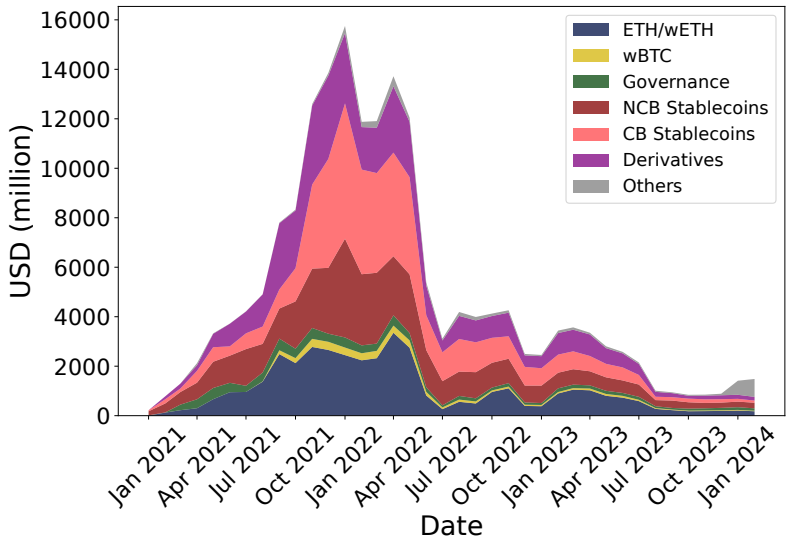


Fig. 6. **Estimated double counting**. The plot shows the evolution in time of the value held in contracts identified by 240 *balanceOf* and *eth_getBalance* queries called on the same contracts and tokens but associated with different protocols (see Subsection 3.5), potentially contributing to double counting.

4.3 Estimation of contract-level double counting

The approach used to compute verifiable Total Value Locked can also be leveraged to quantify the economic magnitude of the double counting issue identified at the infrastructure level, arising when a smart contract included in the TVL calculation is not exclusively associated with one project.

Figure 6 reports thus the evolution over time of the value held in the contract accounts identified in Subsection 3.5 for the balance queries that are repeated over multiple protocols, thus potentially contributing to double counting. Compared to the TVL of major DeFi protocols, it is composed of a higher share of derived tokens and crypto-backed stablecoins. Notably, the value reached a peak of almost 16bln$ at the end of 2021, coinciding with an all-time high of DeFi TVL and a period of intense reliance on such figures for investment and governance decisions. Double counting is therefore a threat to a correct interpretation of the TVL metric also at the infrastructure level, and its potential impact is economically relevant.

4.4 Composition of value locked in DeFi

We now examine the composition of verifiable total value locked and how it evolved over time. As Figure 7 shows, wETH maintained its leading position throughout the period, with dominance values varying from 56.77% in Jan 2021 to 23% in Jul 2022, and 44.8% in Jan 2024. Stablecoins like USDT, USDC, and BUSD competed for the second place, each constituting from around 5% to 20% of vTVL over the time period. Liquid staking token stETH emerged steadily to claim a top-5 position by 2024, while wBTC maintained a more stable position among the first tokens over time. The remaining positions are occupied by other tokens, including governance and crypto-backed stablecoins, with a lower share of 1 to 2 percentage points.

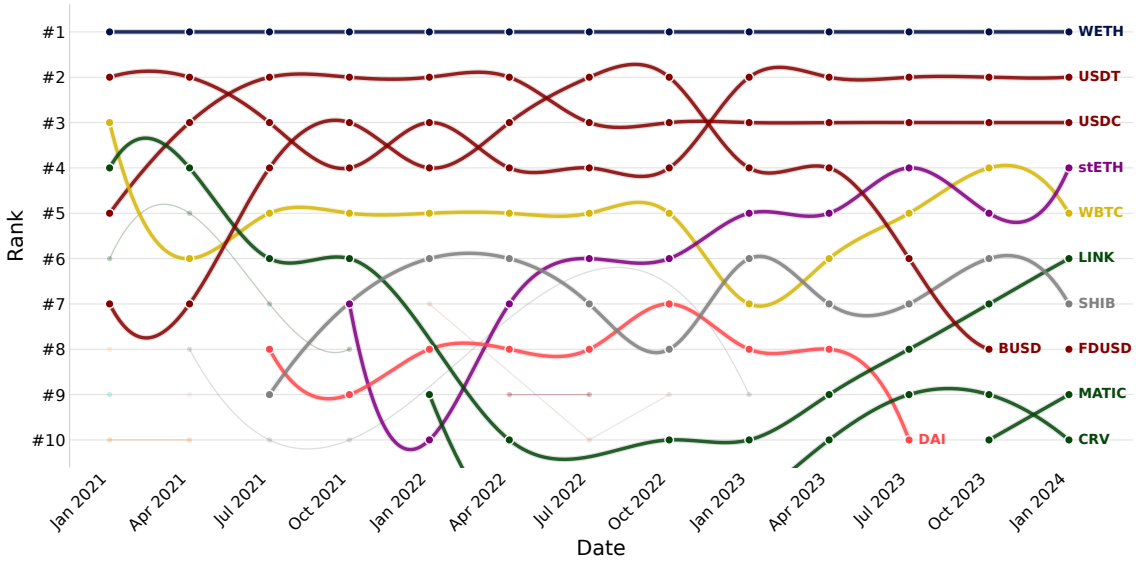


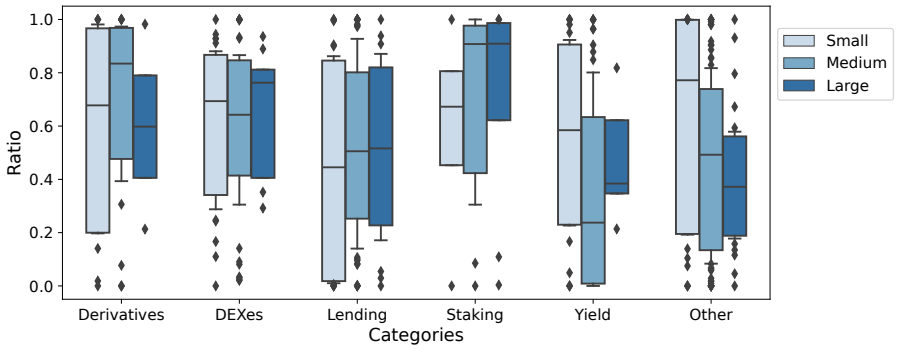
Fig. 7. **Token Dominance Shifts in vTVL**. The plot shows the top 10 cryptoasset ranking by value locked, from January 2021 to January 2024. Note that the time series labeled as wETH includes both wrapped Ether and Ether.

Finally, we analyze how the ratio between the value of non-derived assets (as defined in Luo et al. [36]) and the total value locked, $R = \frac{ND_a}{TVL}$, changes across protocol category, protocol size, and time. For this analysis, we utilize the DeFiLlama published figures to avoid data loss and utilize our token categorization to compute the amount of each protocol as the US dollar amount of plain tokens, i.e., native tokens, governance tokens, and NCB stablecoins. We focus on the subset of protocols with average TVL larger than $5 \cdot 10^4$ USD to ensure noise reduction and group them in DeFi categories as reported in Table 12. The median value of R is respectively 69.1%, 79.0%, 49.2%, 84.8%, 36.1%, and 48.1% for DEXes, Derivatives, Lending, Staking, Yield, and Other protocols. Yield protocols highly rely on non-redeemable tokens, while Derivatives and Staking protocols have the highest ratio, indicating lower reliance on them.

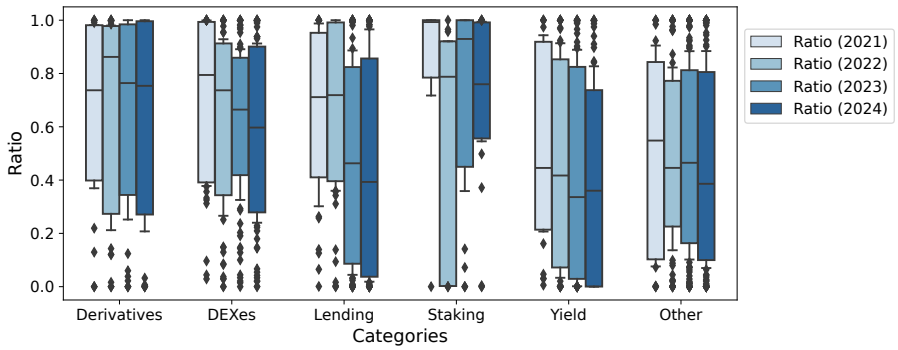
Figure 8 shows two panels conveying additional information on the ratio R, with protocols further divided by size (upper panel) and time (lower panel). More precisely, on top we divide protocols in small size (N = 117), medium size (N = 232), and large size (N = 63), respectively when they have average TVL below 10^6 USD, between 10^6 USD and 10^8 USD, and larger than 10^8 USD. On the bottom, we split the R values for each protocol across time (2021, 2022, 2023, 2024). Interestingly, we do not find strong patterns when distinguishing protocols by size. Instead, in Panel (b) we observe that for DEXes and Lendingprotocols, the median of R is steadily decreasing over time, indicating increased reliance on derived tokens.

5 A Composite Verifiability Score

The analyses conducted in Sections 3 and 4 show a heterogeneous picture of TVL computation in practice: protocols differ substantially in their reliance on off-chain sources, their use of non-standard balance functions, their potential contribution to double counting, and the degree to which their reported figures can be independently verified. To synthesize these dimensions into a single, protocol-level summary, we introduce a composite verifiability score.



(a)



(b)

Fig. 8. **Ratio of Total Value Redeemable over Total Value Locked.** Protocols are grouped by category and further divided by size (a) and time (b). The y-axis indicates the ratio R between TVR and TVL. DEXes and Lending protocols rely more on non-redeemable tokens in 2024 w.r.t. earlier years.

To do so, we aggregate five binary indicators, each capturing one of the five key dimensions identified in our study. The first indicator equals 1 if the protocol produces at least one on-chain call and no calls to external hosts, and 0 otherwise (including thus protocols that combine on-chain and off-chain calls). The second equals 1 if the protocol relies exclusively on standard balance queries (balanceOf and eth_getBalance), and 0 if alternative functions are used or if on-chain interactions could not be verified at all. The third equals 1 if no double-counting issue at the contract level was detected for the protocol - that is, none of its balance queries appear shared with another protocol on the same contract and token - and 0 if double counting was found or could not be ruled out. The fourth equals 1 if none of the protocol's balance queries target a homonymous contract impersonating a well-known token (Section 3.6), and 0 if such a query was detected or could not be ruled out owing to off-chain reliance. The fifth equals 1 if the protocol's Discrepancy Ratio falls within ±0.5, indicating that our on-chain reconstruction aligns with published figures, and 0 otherwise. The score ranges thus from 0 to 5, with higher values indicating greater verifiability, reproducibility, and transparency of the underlying project; we group protocols into low (0–1), mid (2–3), and high (4–5) verifiability bands.

	Low		Medium		High	
Score	0	1	2	3	4	5
Share	13.95%	4.69%	2.66%	26.52%	36.74%	15.44%

Table 3. **Score ranking aggregated**.

5.1 Verifiability score: results

Table 3 reports the resulting distribution across the 939 protocols in our dataset, showing the proportion of protocols associated with each value in the range. Grouping protocols into the three bands defined above, 52.2% fall in the high band (score 4–5), 29.2% in the mid band (2–3), and 18.6% in the low band (0–1). A majority of protocols falls into the highest band, suggesting that the ecosystem is already leaning toward the adoption of verifiable TVL computation methods. At the same time, however, the 18.6% of protocols scoring 0 or 1 should not be overlooked: these capture cases where off-chain dependencies, undocumented balance functions, or unresolvable double-counting issues make independent verification effectively not possible. Taken together, these results suggest that while a verifiability-oriented culture is rooted in the DeFi ecosystem, structural and methodological barriers remain for a significant minority of protocols. These observations motivate the design guidelines we discuss below.

6 Discussion: Towards TVL Standardization

TVL today remains largely non-standardized and its computation open source to third parties. Against this backdrop, informed by the challenges identified in TVL computation and the case study conducted, we discuss a series of recommendations that may help guide the design of reproducible, verifiable, transparent and interpretable TVL estimations.

First, **TVL can and should be computed from available on-chain sources**. As discussed in Sections 3 and 4, relying on external services may hinder verifiability and reproducibility, raise concerns about transparency and create opportunities for manipulation or double counting. Furthermore, third parties need to **know explicitly what protocol-specific smart contracts and associated tokens** are utilized to count value. This also allows to verify if the association of a contract to a project is unique or a potential source of double counting.

The use of **standard balance methods should be preferred over custom functions whenever possible**, in order to standardize computation methods and enhance interpretability of the metric. As discussed in Section 3.1, the most common tokens are ERC-20 compliant and therefore it is straightforward to understand how their balance is quantified. If tokens are non-standard or rely on alternative functions, proper documentation is required to interpret results. A promising finding in this sense is the increased predominance over time of standard functions, which revealed a trend towards homogenization.

We propose that aggregators **publish their token categorizations** and enable third parties to independently **select which assets to include or exclude in computations**, with particular attention to derived and homonym tokens that mimic the name or symbol of well-known ones. While aggregators today allow to include or exclude certain tokens from computations, such as governance tokens, borrowed tokens in lending protocols, and others [9], we could not find a detailed list of what assets are included in each category. Furthermore, as indicated in previous research [36], derivative tokens are responsible for double counting, and this aspect needs to be taken into account. In our case study, we grouped tokens into categories and identified derivative tokens to provide deeper insights on TVL composition,

allowing direct control on the assets selected for computation. Finally, token categorizations should also flag tokens that mimic the name or symbol of widely traded tokens, since reliance on homonym tokens points to either misconfiguration or deliberate deception and opens room for value inflation.

More broadly, it is also necessary to **define common standards for protocol selection criteria** and maintain a public list with included protocols. According to DeFiLlama APIs, the protocols we analyzed are grouped into 39 categories: the most common ones are DEXes, Yield, Lending, and Services (respectively $N = 78, 68, 55, 34$). Notably, $N = 28$ are labeled as CEXs. It is also arguable whether other categories belong to DeFi (Gaming, Oracle, Wallets) or what their purpose is (SoFi, Launchpad, Chain). Furthermore, relying on self-reporting also implies self-selection bias to a certain extent: some protocols might not be interested in reporting data to all TVL aggregators and viceversa. Related to protocol selection, it is important to **complement protocol-specific information with metadata for version management**. From the analysis in Section 4.1, it emerged that protocols plugins do not manage versions consistently. As protocols often deploy new versions, it is important to have a clear vision of what funds are deposited in each protocol version.

We acknowledge that future research might address some limitations of our study. First, our work is limited to the DeFiLlama infrastructure and to Ethereum alone, but DeFi is spread across DLTs; whilst we believe our findings can be generalized to other blockchains that replicate Ethereum, future analyses could pay specific attention to bridges and multi-chain protocols, and include other blockchains like Solana. Second, alternative computation methods to standard ones should be investigated in detail, to understand what functionalities they offer and why they are used. Similarly, the collection of token prices can be improved. We could not price all tokens in our dataset with our approach (but we did cover all the most relevant ones), and on-chain prices should be extracted from multiple DEXes and liquidity pools to account for low-liquidity scenarios. The price of certain tokens, like NFTs, might need to be complemented with exchange market price data. Third, relying only on on-chain sources is likely more costly and less efficient computationally. This approach might face resistance due to the varied nature of the platforms involved. Following the suggested approach, protocols might have to disclose more information than they do today. As this comes at a gain in terms of transparency and verifiability, it is important to further investigate how to balance these aspects.

Beyond our analysis, we note that TVR [36] is primarily devised for application on an entire ecosystem and therefore helps standardize TVL at the ecosystem level. However, it remains still unclear how to conduct asset selection at the protocol level. Appendix A.6 provides further analyses on TVL composition changes in relation to TVR and across protocol category, size, and time. Further research should focus on this aspect.

Finally, we remark that the verifiability and transparency of TVL computations is part of a broader discussion on DeFi-related risks and issues. Indeed, DeFi automation and the removal of human involvement has introduced or heightened certain risks, e.g. diminishing oversight and control, or paving the way for new types of intermediaries [5]. Just as TVL should be verifiable, proof-of-reserve systems are essential for stablecoins and cryptoasset trading platforms [22, 44]. Moreover, the need for governance makes some degree of centralization unavoidable [21], and structural aspects of the system contribute to the concentration of power in the hands of few developers [28, 33].

7 Conclusions

In this work, we conduct a systematic study on 939 DeFi projects deployed in Ethereum. We first provide a comprehensive understanding of the methodologies currently used for TVL computation; next, we examine the extent to which TVL is reproducible and verifiable using available on-chain data by introducing a new metric, the verifiable Total Value Locked (vTVL). Informed by these analyses, we propose design guidelines and possible standardization attempts in the field.

TVL is a fundamental financial metric in the DeFi ecosystem. Reaching common standards and publishing verifiable figures is critical to obtaining a clear overview of the true dimensions of the DeFi ecosystem, informing correctly users' investment decisions, and guaranteeing fair competition across protocols. This also supports the need for greater financial transparency and accountability to address the expectations of diverse stakeholder groups, including users, investors, and regulators. This work provides several insights in this direction.

References

[1] 4byte. [n. d.]. Ethereum Signature Database. See https://www.4byte.directory.

[2] Raphael Auer, Bernhard Haslhofer, Stefan Kitzler, Pietro Saggese, and Friedhelm Victor. 2024. The technology of decentralized finance (DeFi). *Digital Finance* 6, 1 (2024), 55–95.

[3] Matt Brigida. 2025. The surprising irrelevance of total-value-locked on cryptocurrency returns. *Economics Letters* (2025), 112673.

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

[5] Nic Carter and Linda Jeng. 2021. DeFi protocol risks: The paradox of DeFi. *Regtech, suptech and beyond: innovation and technology in financial services" riskbooks–forthcoming* 3 (2021).

[6] Faris Chaudhry. 2025. Relationship between total value locked and the yield dynamics of liquid restaking tokens. In *2025 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 1–7.

[7] Jonathan Chiu, Thorsten Koeppl, Hanna Yu, and Shengxing Zhang. 2023. *Understanding DeFi Through the Lens of a Production-Network Model*. Technical Report. Bank of Canada.

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

[9] Coindesk. 2022. Data Provider DeFiLlama De-emphasizes Double-Counted Crypto Deposits After Saber Revelation. Available at: https://www.coindesk.com/business/2022/08/05/data-provider-defillama-de-emphasizes-double-counted-crypto-deposits-after-saber-revelation/.

[10] Coingecko. [n. d.]. Top Crypto Categories By Market Cap. Available at: https://www.coingecko.com/en/categories.

[11] Coingecko. [n. d.]. What Total Value Locked (Tvl) and Why Users Monitor This Metric. See: https://www.coingecko.com/learn/total-value-locked.

[12] Coinmarketcap. [n. d.]. Cryptocurrency Sectors by 24h Price Change. Available at: https://coinmarketcap.com/cryptocurrency-category/.

[13] Simon Cousaert, Jiahua Xu, and Toshiko Matsui. 2022. Sok: Yield aggregators in defi. In *2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 1–14.

[14] DappRadar. [n. d.]. DeFi Rankings. Available at: https://docs.dappradar.com/rankings/defi-rankings.

[15] DeFiLlama. [n. d.]. APIs. Available at: https://defillama.com/docs/api.

[16] DeFiLlama. [n. d.]. DeFiLlama website. Available at: https://defillama.com/.

[17] DeFiLlama. [n. d.]. GitHub repository - DeFiLlama-Adapters. Available at: https://github.com/DefiLlama/DefiLlama-Adapters.

[18] DeFiLlama. [n. d.]. Why we don't accept APIs. Available at: https://github.com/DefiLlama/DefiLlama-Adapters/discussions/432.

[19] DeFiPulse. [n. d.]. DeFiPulse Total Value Locked (TVL) Methodology. Available at: https://docs.defipulse.com/methodology/tvl.

[20] Monika Di Angelo and Gernot Salzer. 2023. Identification of token contracts on Ethereum: standard compliance and beyond. *International Journal of Data Science and Analytics* 16, 3 (2023), 333–352.

[21] Jon Frost Doerr, Anneke Kosse, Asad Khan, Ulf Lewrick, Benoît Mojon, Benedicte Nolens, and Tara Rice. 2021. DeFi risks and the decentralisation illusion. *BIS Quarterly Review* 21 (2021).

[22] Barry Eichengreen, My T Nguyen, and Ganesh Viswanath-Natraj. 2023. Stablecoin Devaluation Risk. *WBS Finance Group Research Paper* (2023). Available at SSRN: http://dx.doi.org/10.2139/ssrn.4460515.

[23] Ethereum.org. [n. d.]. ERC-20: Token Standard Improvement Proposal. See https://eips.ethereum.org/EIPS/eip-20.

[24] Ethereum.org. [n. d.]. Ethereum JSON-RPC API. See https://ethereum.org/en/developers/docs/apis/json-rpc/#eth_getbalance.

[25] Etherscan. [n. d.]. AaveV2Provider contract.

[26] Etherscan. [n. d.]. Vyper contract. See https://etherscan.io/address/0xba3cfea6514cf5acddeff3167df0b7a4337751bc#code.

[27] Sizheng Fan, Tian Min, Xiao Wu, and Cai Wei. 2022. Towards understanding governance tokens in liquidity mining: a case study of decentralized exchanges. *World Wide Web* (2022), 1–20.

[28] Cesare Fracassi, Moazzam Khoja, and Fabian Schär. 2024. Decentralized crypto governance? transparency and concentration in ethereum decision-making. *Transparency and Concentration in Ethereum Decision-Making* (2024).

[29] Michael Fröwis, Andreas Fuchs, and Rainer Böhme. 2019. Detecting token systems on ethereum. In *International Conference on Financial Cryptography and Data Security*. Springer, 93–112.

[30] 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.

[31] Lioba Heimbach, Ye Wang, and Roger Wattenhofer. 2021. Behavior of liquidity providers in decentralized exchanges. *arXiv preprint arXiv:2105.13822* (2021).

[32] Tamás Katona. 2021. Decentralized finance: the possibilities of a blockchain "money lego" system. *Financial and Economic Review* 20, 1 (2021), 74–102.

[33] Stefan Kitzler, Stefano Balietti, Pietro Saggese, Bernhard Haslhofer, and Markus Strohmaier. 2024. The governance of decentralized autonomous organizations: A study of contributors' influence, networks, and shifts in voting power. In *International Conference on Financial Cryptography and Data Security*. Springer, 313–330.

[34] Stefan Kitzler, Friedhelm Victor, Pietro Saggese, and Bernhard Haslhofer. 2023. Disentangling decentralized finance (DeFi) compositions. *ACM Transactions on the Web* 17, 2 (2023), 1–26.

[35] 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*. 155–166.

[36] Yichen Luo, Yebo Feng, Jiahua Xu, and Paolo Tasca. 2025. Piercing the Veil of TVL: DeFi Reappraised. In *International Conference on Financial Cryptography and Data Security*. Springer.

[37] Youcef Maouchi, Lanouar Charfeddine, and Ghassen El Montasser. 2022. Understanding digital bubbles amidst the COVID-19 pandemic: Evidence from DeFi and NFTs. *Finance Research Letters* 47 (2022), 102584.

[38] Dominik Metelski and Janusz Sobieraj. 2022. Decentralized Finance (DeFi) Projects: A Study of Key Performance Indicators in Terms of DeFi Protocols' Valuations. *International Journal of Financial Studies* 10, 4 (2022), 108.

[39] Amani Moin, Kevin Sekniqi, and Emin Gun Sirer. 2020. SoK: A classification framework for stablecoin designs. In *International Conference on Financial Cryptography and Data Security*. Springer, 174–197.

[40] Danny Nelson and Tracy Wang. [n. d.]. Master of Anons: How a Crypto Developer Faked a DeFi Ecosystem. https://www.coindesk.com/layer2/2022/08/04/master-of-anons-how-a-crypto-developer-faked-a-defi-ecosystem/.

[41] Lucas Nuzzi, Antoine Le Calvez, and Kyle Waters. 2021. Understanding Total Value Locked (TVL). Available at: https://coinmetrics.substack.com/p/coin-metrics-state-of-the-network-0c0.

[42] Kanis Saengchote et al. 2021. Where Do DeFi Stablecoins Go? A Closer Look at What DeFi Composability Really Means. *Available at SSRN 3893487* (2021).

[43] Pietro Saggese, Michael Fröwis, Stefan Kitzler, Bernhard Haslhofer, and Raphael Auer. 2025. Towards verifiability of total value locked (TVL) in decentralized finance. In *2025 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 1–9.

[44] Pietro Saggese, Esther Segalla, Michael Sigmund, Burkhard Raunig, Felix Zangerl, and Bernhard Haslhofer. 2024. Assessing the solvency of virtual asset service providers: are current standards sufficient? *Applied Economics* (2024), 1–16.

[45] Kirill Shilov and Andrey Vitalievich Zubarev. [n. d.]. Return Factors of Ether Cryptocurrency: On Chain Metrics and Defi. *Available at SSRN 4432586* ([n. d.]).

[46] Florentina Şoiman, Guillaume Dumas, and Sonia Jimenez-Garces. 2022. The return of (I) DeFiX. *arXiv preprint arXiv:2204.00251* (2022).

[47] Stelareum. [n. d.]. Total Value Locked (TVL) in DeFi protocols. Available at: https://www.stelareum.io/en/defi-tvl.html.

[48] Viktorija Stepanova and Ingars Eriņš. 2021. Review of decentralized finance applications and their total value locked. *TEM Journal* 10, 1 (2021), 327.

[49] Uniswap. [n. d.]. Factory contract documentation. See https://docs.uniswap.org/contracts/v2/reference/smart-contracts/factory.

[50] Uniswap. [n. d.]. Functionalities documentation. See https://docs.uniswap.org/contracts/v2/reference/smart-contracts/pair.

[51] Wenbin Wu. 2026. Tokens All the Way Down: A Money View of Decentralized Finance. *arXiv preprint arXiv:2603.01803* (2026).

[52] Xihan Xiong, Zhipeng Wang, Xi Chen, William Knottenbelt, and Michael Huth. 2023. Leverage staking with liquid staking derivatives (lsds): Opportunities and risks. *arXiv preprint arXiv:2401.08610* (2023).

[53] Jiahua Xu, Krzysztof Paruch, Simon Cousaert, and Yebo Feng. 2023. Sok: Decentralized exchanges (dex) with automated market maker (amm) protocols. *Comput. Surveys* 55, 11 (2023), 1–50.

[54] Teng Andrea Xu and Jiahua Xu. 2022. A short survey on business models of decentralized finance (DeFi) protocols. In *International Conference on Financial Cryptography and Data Security*. Springer, 197–206.

[55] Liyi Zhou, Xihan Xiong, Jens Ernstberger, Stefanos Chaliasos, Zhipeng Wang, Ye Wang, Kaihua Qin, Roger Wattenhofer, Dawn Song, and Arthur Gervais. 2023. Sok: Decentralized finance (defi) attacks. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2444–2461.

A Appendix

A.1 Reproducibility and reliance on off-chain data

In this appendix, we provide additional information on the servers and errors that were detected during TVL computation. Table 4 reports a list of the documented errors categorized by typology. The most occurring ones are related to the block height provided not being accepted, or are caused by a collection of asynchronous operations that did not complete successfully. Other errors can be reconducted to the lack of missing fields or parameters and other technical problems. Table 5 reports a full list of the detected servers. While the one occurring most frequently is part of TheGraph infrastructure, an open-source software used to collect, process, and store data from various blockchain applications, the majority are servers from third parties. We note that we excluded the server 'coins.llama.fi', as it is likely associated with DeFiLlama itself and used for internal operations.

Error	Count
Block height	53
Asynchronous calls failed	17
Key required	5
Missing field/parameter	5
Undefined/null object	4
Call method failed	3
Invalid token/balance	2
GraphQL error	1

Table 4. **Errors occurred during TVL computation.** Most of them are related to technical issues when running the TVL-computing code.

A.2 Heterogeneity of on-chain interactions

We now discuss the network analysis results for the frequency of occurrence in absolute terms of the functions called, of the number of protocols calling each specific function, and for the number of token calls in *balanceOf* functions. Following established methodologies [4, 8], we estimate the parameters $\hat{\theta} = (\hat{k}_{min}, \hat{\alpha})$ and conduct a goodness-of-fit test via a bootstrapping procedure (N = 1,000). The resulting p-value indicates if the power law is a plausible fit for the empirical data (i.e., $p \geq 0.1$) or not. A log-likelihood ratio ($\mathcal{R}$) test is conducted to compare the power-law fit against other heavy-tailed distributions (exponential, lognormal, and weibull). While the bootstrap analysis shows that the hypothesis that a power-law distribution is a good fit holds only for the distribution of the number of protocols calling each specific function, in all three cases the power law is either a better fit with respect to the other distributions, or the test is inconclusive. An inflection point in the distribution of the number of token calls in *balanceOf* functions, between the values 400 and 1,000 of the x-axis, indicates that values on the right of the elbow are overrepresented and potentially the existence of a transition region.

Next, we comment the approach used to identify alternative functions likely used to compute TVL. Table 6 reports the most called on-chain functions in absolute terms, rather than being ranked by the number of protocols that call them. We notice that a number of function names (*balanceOfUnderlying*, *balance*, as well as *totalAssets* that appeared in Table 1), indicate functions that are likely to compute balance in alternative ways with respect to the *balanceOf* most common method. To identify all these functions, we use a set of regular expressions that capture any method whose function name includes the term 'balance', or a case-insensitive combination of the following terms: 'total, get, locked' AND 'tvl,

Server	Count	Server	Count
api.thegraph.com	29	config.rampdefi.com	1
raw.githubusercontent.com	5	analytics.back.popsicle.finance	1
sushi-analytics.onrender.com	3	api.affinedefi.com	1
rpc.ankr.com	3	bridge.orbitchain.io	1
vault-content-api.teahouse.finance	2	api.angle.money	1
tvl-adapter-cache.s3.eu-central-1.amazonaws.com	2	api.axelarscan.io	1
api.myso.finance	2	api.beefy.finance	1
crucible.alchemist.wtf	1	api.clipper.exchange	1
data.cian.app	1	api.cream.finance	1
devapi.ease.org	1	api.daomaker.com	1
knit-admin.herokuapp.com	1	api.debridge.finance	1
explorer.poly.network	1	api.defiedge.io	1
f8wgg18t1h.execute-api.us-west-1.amazonaws.com	1	api.exchange.coinbase.com	1
files.insurace.io	1	api.flashstake.io	1
gateway-arbitrum.network.thegraph.com	1	api.flokifi.com	1
counterstake.org	1	api.goldsky.com	1
graph-node.mainnet.termfinance.io	1	api.hord.app	1
graph-proxy.nftx.xyz	1	api.hotcross.com	1
homora-api.alphafinance.io	1	api.mean.finance	1
messina.one	1	api.multibit.exchange	1
lsd-subgraph.joinstakehouse.com	1	api.nodes-brewlabs.info	1
bsc-dataseed1.defibit.io	1	api.resonate.finance	1
metabase.internal-streamflow.com	1	api.staking.ankr.com	1
midgard.ninerealms.com	1	api.studio.thegraph.com	1
moonbeam.public.blastapi.io	1	api.tokensfarm.com	1
partner-api.stafi.io	1	api.unrekt.net	1
polygon-rpc.com	1	api.vesper.finance	1
preserver.mytokenpocket.vip	1	app.everrise.com	1
stakehouse-subgraph.joinstakehouse.com	1	assets.nabox.io	1
static.optimism.io	1	backend.mochi.fi	1
token-list.solv.finance	1	beaconcha.in	1
universe.staderlabs.com	1	bsc-dataseed.binance.org	1

Table 5. **External servers utilized in TVL computations.** An occurrence is counted each time a protocol interacts with a server. While some pose a smaller threat to verifiability, e.g., TheGraph, others appear to be associated with specific protocols.

Method	Count	N of protocols	Method	Count	N of protocols
balanceOf	102655	641	underlying	1613	43
getLockedTokenAtIndex	44022	1	escrows	1203	1
balanceOfUnderlying	4191	3	token0	1040	94
getCurrentTokens	3436	10	token1	1040	94
getReserves	3016	121	tokenByIndex	875	2
totalSupply	2577	154	balance	874	9
token	2456	24	get_coins	814	2
symbol	1835	64	pool_list	809	1
eth_getBalance	1696	170	getEthBalance	791	7
poolInfo	1672	25	calcTotalValue	773	1

Table 6. **Most called functions in absolute terms.** A number of functions (e.g., *balanceOfUnderlying*, *balance*) have names indicating that they are likely used to compute balance in a non-standard way.

Ether, ETH, stake, underlying, reserve, amount, supply, value, locked, shares, asset, liquidity'. Furthermore, we conduct a manual check to remove functions that are clearly not computing TVL but provide supplementary functionalities, such

Function name	Hex Signature	Function name	Hex Signature
accountedBalance	['0x0937eb54']	getTotalRPLStake	['0x9a206c8e']
allBalances	['0x555b6162']	getTotalReserves	['0x242693d3']
balance	['0xb69ef8a8']	getTotalUnderlying	['0xb40494e5']
balanceOf	['0x00fdd58e', '0x35ee5f87', '0x3656eec2', '0xf7888aec']	getTotalValueInPool	['0xc8ecaf30']
balanceOfUnderlying	['0x3af9e669']	getTrackedAssets	['0xc4b97370']
balances	['0x4903b0d1', '0x065a80d8', '0x8909aa3f', '0x27e235e3']	getTvl	['0xd075dd42']
borrowBalanceStored	['0x95dd9193']	getUnderlying	['0x9816f473']
calcTotalValue	['0xc7de38a6']	getUnderlyingBalances	['0x1322d954']
checkBalance	['0x5f515226']	getUnderlyings	['0xf65baefa']
currencyBalance	['0x5c75347a']	getUnderlyingsAmounts-FromClusterAmount	['0x9bb1bebb']
currentTotalStake	['0xce4843e9']	lockedBalances	['0x0483a7f6']
getAllAssets	['0x2acada4d']	lockedLiquidityOf	['0xd9f96e8d']
getAllStakes	['0x04238994']	lockedStakesOf	['0x1e090f01']
getAssets	['0x67e4ac2c']	lockedSupply	['0xca5c7b91']
getBassets	['0x1d3ce398']	poolBalance	['0x96365d44']
getCacheBalances	['0x4a9d1036']	syncBalance	['0xfd9c652b']
getContractValue	['0xdc82697c']	totalAsset	['0xf9557ccb']
getETHPx	['0xab9aadfe']	totalAssetAmount	['0xfd27152c']
getEthBalance	['0x4d2301cc']	totalAssetBorrow	['0x20f6d07c']
getLockedVestings	['0x344e58d3']	totalAssets	['0x01e1d114']
getPoolAmount	['0x945eb764']	totalBalance	['0xad7a672f']
getPoolTotalValue	['0xf1437c16']	totalBalanceOf	['0x4b0ee02a']
getRawFundBalances	['0xe2e4c60c']	totalETH	['0x36bdee74']
getRawFund-BalancesAndPrices	['0x0d8f8a90']	totalReserve	['0x4c68df67']
getReserveTotalBorrows	['0xe6d18190']	totalReserves	['0x8f840ddd']
getSupply	['0xf77ee79d']	totalSYNCLocked	['0xc3f4d79f']
getSupportedAsset	['0x60a8b18a']	totalStaked	['0x817b1cd2']
getSupportedAssets	['0xe5406dbf']	totalTokenBalanceStakers	['0x1878fbf3']
getSupportedAssetsLength	['0xc0fd22b7']	totalValue	['0xd4c3eea0']
getToken1Balance	['0x5153786b']	total_staked	['0xaf7568dd']
getTotalAmounts	['0xc4a7761e']	underlyingBalance	['0x59356c5c']
getTotalAsset	['0x2768385d']	virtualBalance	['0xdcd2af17']
getTotalBalance	['0x12b58349']	virtualUsdtAccumulatedBalance	['0xd88953b4']
getTotalPooledEther	['0x37cfdaca']	yvCurveFRAXBalance	['0xc645065e']

Table 7. **Alternative functions to balanceOf likely used to compute TVL.** For each function we report the name (1^{st} *column*) and its hex signature (2^{nd} *column*).

as *totalSupply* or *getReserves*. Table 7 reports the full list of alternative functions to *balanceOf* likely used to compute TVL. In total, we remove the following 21 functions: *getAssetInfo*, *getAssetsPrices*, *getAssetsWithState*, *getBNFTAssetList*, *getLiquidityPools*, *getLockedTokenAtIndex*, *getNumLockedTokens*, *getReserveData*, *getReserves*, *getReservesData*, *getReservesList*, *getUnderlyingAsset*, *getUnderlyingOfIBTAddress*, *getUnderlyingPrice*, *getUnderlyingTokenAddress*, *totalShares*, *totalSupply*, *totalUnderlying*, *totalUnderlyingSupply*.

A.3 Changes to computation methods over time

The analysis in Section 3.4 follows the intuition that developers can modify how TVL is computed and that the collection may be dependent on one specific implementation, thus TVL computations might change across time and commits. We thus repeat the data gathering on the following commits:

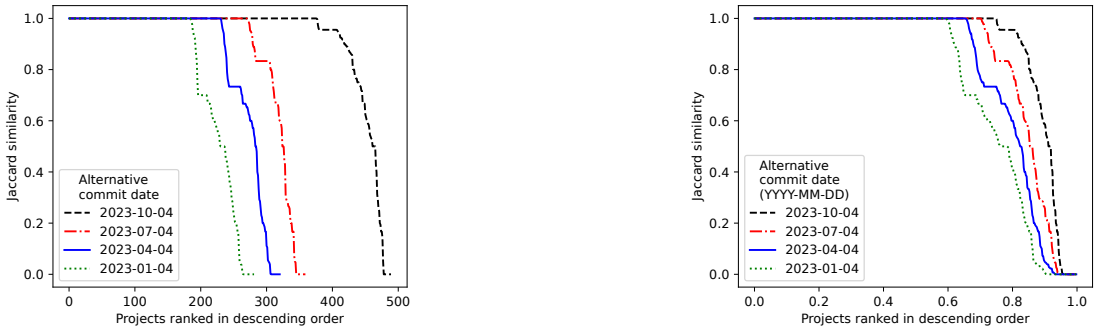
- 6764756f9270ab6a3047c06c13c0b1b2d32a3247: Jan 4 2024;
- aef637c6413b5101667e567a0422769b1ca99564: Oct 4 2023;
- 1bfb7b5c798c7a491694882a7b390ed41385c315: Jul 4 2023;

- 72b764c5d0bb5896f2857dd8c2ced5e89e7fb063: Apr 4 2023;
- 679f52e123a19d9c5160d1aa79a33dd9de6dc5ec: Jan 4 2023;

As discussed in the main body of the paper, we do not analyze data earlier than 2023: the structure of some critical files within the DeFiLlama repository has changed in time, thus making our data-gathering infrastructure less reliable on older commits. In particular, the number of comparable commits decreases steadily with time, and for commits older than Aug 2021, our pipeline becomes not compatible with specific changes made to the repository (see https://github.com/DefiLlama/DefiLlama-Adapters/discussions/432). Table 8 shows that differences in the dataset of captured calls across commits exist: the total amount of calls to an Ethereum node is not stable over time, and few specific protocols play a relevant role in this sense: as one can see in Table 8, the differences are markedly smaller when some specific projects are excluded.

	Jan 24	Oct 23	Jul 23	Apr 23	Jan 23
Total calls	197775	333239	262512	314280	254259
Unicrypt calls	44032	198228	150094	104165	77063
Other calls	153743	135011	112418	210115	177196

Table 8. **Dataset dimension in different commits**. One specific protocol (Unicrypt) is responsible for large variations across commits.



(a) Figure 3a, x-axis non normalized

(b) Figure 3a, all protocols

Fig. 9. **Jaccard similarity.** Panel (a) reports the same information of Figure 3a, but projects are reported on the x-axis in absolute terms instead of being normalized. The sample of comparable protocols decreases sharply when older commits are compared. Panel (b) shows instead the Jaccard similarity values when including also projects that raised errors in the data-gathering process or used external hosts. Average values range from 0.90% to 0.75%.

Next, in Figure 9 we report additional information on the Jaccard similarity analysis. We recall that the Jaccard similarity coefficient, utilized for measuring the similarity of overlapping sets, is defined as the ratio between their intersection and their union, and ranges between 0 and 1. We therefore favor it over alternative similarity metrics that measure distances between text strings (e.g., Hamming), vectors (e.g., cosine similarity) or that identify correlations across data points (e.g., Spearman or Pearson). The left panel shows the same information of Figure 3a, but the x-axis is not normalized. The right panel includes instead all protocols, therefore also the ones that raised errors in the data-gathering process or used external hosts. The average Jaccard values are 0.90%, 0.85%, 0.80%, and 0.75% respectively

for Oct, Jul, Apr, and Jan 2023. Notably, we cannot exclude that differences in older commits are due to a less reliable collection. Therefore, our findings can be interpreted as a lower boundary for the similarity values.

Finally, Table 9 reports information on the evolution over time of the ratio between the number of standard and alternative balance queries (excluding protocols that used hosts or raised errors during the data collection).

	Commit identifiers and date				
	676475 (2024 01-04)	aef637 (2023 10-04)	1bfb7b (2023 07-04)	72b764 (2023 04-04)	679f52 (2023 01-04)
Alt. functions	6831	7554	6836	6882	7197
Std. balance queries	70073	54144	27534	23426	18367
Ratio	0.089	0.122	0.199	0.227	0.282

Table 9. **Evolution over time of the ratio between the number of standard and alternative balance queries.** Standard balance queries include *balanceOf* and *eth_getBalance* calls.

A.4 Non-mutually exclusive smart contract calls

Input	Protocols
0x3980c9ed79d2c191a89e02fa3529c60ed6e9c04b	['metis' 'metisBridge']
0x8301ae4fc9c624d1d396cbdaa1ed877821d7c511	['curve' 'bent']
0xb576491f1e6e5e62f1d8f26062ee822b40b0e0d4	['curve' 'bent']
0xd51a44d3fae010294c616388b506acda1bfaae46	['curve' 'bent']
0xdc24316b9ae028f1497c275eb9192a3ea0f67022	['curve' 'bent']

Table 10. **Duplicated get_ethBalance functions**. get_ethBalance calls executed by different protocols on the same input address (column 'Input').

We report the list of duplicated addresses in Tables 10 and 11. These correspond to calls executed by different protocols (column 'Protocols') on the same input address (column 'Input') and token address (only for Table 11, column 'On'). Upon closer inspection, we find that in most cases they are related to interconnected protocols (see, e.g., metis and metisBridge and Curve and Bent). One possible explanation for this phenomenon is that these addresses are reported directly by the project developers; we cannot exclude that some of the smart contracts are incorrectly reported multiple times only for a temporary time span and that they are further removed by the maintainers of the DeFiLlama platform.

Input	On	Protocols	Input	On	Protocols
0x031816fd...	0x03e173ad...	['dodo' 'thales']	0x031816fd...	0xc02aaa39...	['dodo' 'thales']
0x0f41eade...	0xc36442b4...	['pawnfi-lending' 'pawnfi-nft']	0x16770d64...	0xc02aaa39...	['enzyme' 'diva']
0x19b080fe...	0x96e61422...	['keep3r' 'curve']	0x1a26ef65...	0x42bbfa2e...	['bobagateway' 'boba']
0x1a26ef65...	0xd26114cd...	['bobagateway' 'boba']	0x1ce8aafb...	0xae7ab965...	['enzyme' 'diva']
0x23012599...	0xba30e5f9...	['pawnfi-lending' 'pawnfi-nft']	0x25d6fe0d...	0x49cf6f5d...	['pawnfi-lending' 'pawnfi-nft']
0x27e49962...	0x790b2cf2...	['pawnfi-lending' 'pawnfi-nft']	0x27f23c71...	0xc02aaa39...	['enzyme' 'nexus']
0x306b1950...	0xeca82185...	['uma' 'perlinx']	0x325a0e5c...	0xae7ab965...	['swell-vault' 'enzyme']
0x325a0e5c...	0xc02aaa39...	['swell-vault' 'enzyme']	0x32ecc1de...	0xb7f7f6c5...	['pawnfi-lending' 'pawnfi-nft']
0x3980c9ed...	0x1f9840a8...	['metis' 'metisBridge']	0x3980c9ed...	0x2260fac5...	['metis' 'metisBridge']
0x3980c9ed...	0x3405a1bd...	['metis' 'metisBridge']	0x3980c9ed...	0x4fabb145...	['metis' 'metisBridge']
0x3980c9ed...	0x51491077...	['metis' 'metisBridge']	0x3980c9ed...	0x6226e00b...	['metis' 'metisBridge']
0x3980c9ed...	0x6b175474...	['metis' 'metisBridge']	0x3980c9ed...	0x6b359506...	['metis' 'metisBridge']
0x3980c9ed...	0x7fc66500...	['metis' 'metisBridge']	0x3980c9ed...	0x9e32b13c...	['metis' 'metisBridge']
0x3980c9ed...	0xa0b86991...	['metis' 'metisBridge']	0x3980c9ed...	0xba6b0dbb...	['metis' 'metisBridge']
0x3980c9ed...	0xd533a949...	['metis' 'metisBridge']	0x3980c9ed...	0xdac17f95...	['metis' 'metisBridge']
0x3a93e863...	0xeca82185...	['uma' 'perlinx']	0x3e75dcad...	0xa0b86991...	['domfi' 'uma']
0x3f1b0278...	0xfafdf0c4...	['keep3r' 'curve']	0x41284a88...	0x0bc529c0...	['percent' 'balancer-v1']
0x41284a88...	0xc02aaa39...	['percent' 'balancer-v1']	0x43b4fdfd...	0xbc6da0fe...	['curve' 'bent']
0x46f5e363...	0xeca82185...	['uma' 'perlinx']	0x4a2f0ca5...	0x1f573d6f...	['bancor' 'ichifarm']
0x4a2f0ca5...	0x903bef17...	['bancor' 'ichifarm']	0x4e8d60a7...	0xc02aaa39...	['degenerative' 'uma']
0x4f1424ce...	0xa0b86991...	['degenerative' 'uma']	0x505efcc1...	0x04abeda2...	['nest' 'parasset']
0x516f5959...	0xc02aaa39...	['degenerative' 'uma']	0x55a8a39b...	0x99d8a9c4...	['curve' 'bent']
0x55a8a39b...	0xa47c8bf3...	['curve' 'bent']	0x58378f5f...	0x903bef17...	['ichifarm' 'balancer-v1']
0x58378f5f...	0xc02aaa39...	['ichifarm' 'balancer-v1']	0x5a6a4d54...	0x99d8a9c4...	['curve' 'bent']
0x5eeaef7d...	0xed5af388...	['pawnfi-lending' 'pawnfi-nft']	0x5f0a4a59...	0xbc4ca0ed...	['pawnfi-lending' 'pawnfi-nft']
0x7514799c...	0xe012baf8...	['pawnfi-lending' 'pawnfi-nft']	0x799c9518...	0xa0b86991...	['degenerative' 'uma']
0x7c62e5c3...	0xc02aaa39...	['degenerative' 'uma']	0x7d0b6fb1...	0x60e4d786...	['pawnfi-lending' 'pawnfi-nft']
0x82c427ad...	0xc02aaa39...	['opyn-squeeth' 'uniswap']	0x8301ae4f...	0xc02aaa39...	['curve' 'bent']
0x8301ae4f...	0xd533a949...	['curve' 'bent']	0x8461a004...	0x95dfdc81...	['keep3r' 'curve']
0x8818a9bb...	0x5555f75e...	['keep3r' 'curve']	0x94e653af...	0xa0b86991...	['domfi' 'uma']
0x99e58237...	0x6b175474...	['percent' 'balancer-v1']	0x99e58237...	0xc02aaa39...	['percent' 'balancer-v1']
0x9a5c88ac...	0x04abeda2...	['nest' 'parasset']	0x9c2c8910...	0x1cc481ce...	['keep3r' 'curve']
0x9d046499...	0x62b9c735...	['curve' 'bent']	0x9d046499...	0xd533a949...	['curve' 'bent']
0x9fe9bb6b...	0x9ea3b5b4...	['delta' 'core']	0xaa5a67c2...	0x86537736...	['curve' 'bent']
0xaff95ac1...	0x903bef17...	['rari' 'ichifarm']	0xb1a3e5a8...	0xa0b86991...	['degenerative' 'uma']
0xb39edbc5...	0xe0a97733...	['tokensfarm' 'bloxmove']	0xb40ba947...	0xeca82185...	['uma' 'perlinx']
0xb576491f...	0x4e3fbd56...	['curve' 'bent']	0xb576491f...	0xc02aaa39...	['curve' 'bent']
0xba3436fd...	0x1a7e4e63...	['angle' 'curve']	0xba3436fd...	0x1abaea1f...	['angle' 'curve']
0xbaaa1f5d...	0x853d955a...	['curve' 'bent']	0xbaaa1f5d...	0x956f47f5...	['curve' 'bent']
0xbaaa1f5d...	0xbc6da0fe...	['curve' 'bent']	0xbebc4478...	0x6b175474...	['curve' 'bent']
0xbebc4478...	0xa0b86991...	['curve' 'bent']	0xbebc4478...	0xdac17f95...	['curve' 'bent']
0xc3160c5c...	0x40803cea...	['spool-v2' 'spool']	0xc48b8329...	0xc00e94cb...	['percent' 'balancer-v1']
0xc48b8329...	0xc02aaa39...	['percent' 'balancer-v1']	0xc697051d...	0x7fc66500...	['aave' 'balancer-v1']
0xc697051d...	0xc02aaa39...	['aave' 'balancer-v1']	0xca2531b9...	0xc02aaa39...	['degenerative' 'uma']
0xceaf7747...	0xa693b19d...	['curve' 'bent']	0xd3a0e00f...	0xa0b86991...	['domfi' 'uma']
0xd50fbace...	0xeca82185...	['uma' 'perlinx']	0xd51a44d3...	0x2260fac5...	['curve' 'bent']
0xd51a44d3...	0xc02aaa39...	['curve' 'bent']	0xd51a44d3...	0xdac17f95...	['curve' 'bent']
0xd632f226...	0x6c3f90f0...	['fraxfinance' 'bent']	0xd632f226...	0x853d955a...	['curve' 'bent']
0xd6ac1cb9...	0x69681f8f...	['keep3r' 'curve']	0xdc24316b...	0xae7ab965...	['curve' 'bent']
0xdcef968d...	0xa0b86991...	['curve' 'fraxfinance']	0xe010fcda...	0x51491077...	['percent' 'balancer-v1']
0xe010fcda...	0xc02aaa39...	['percent' 'balancer-v1']	0xe867be95...	0xba100000...	['percent' 'balancer-v1']
0xe867be95...	0xc02aaa39...	['percent' 'balancer-v1']	0xe969991c...	0xa0b86991...	['percent' 'balancer-v1']
0xe969991c...	0xc02aaa39...	['percent' 'balancer-v1']	0xeb85b2e1...	0xbc16da9d...	['percent' 'balancer-v1']
0xeb85b2e1...	0xc02aaa39...	['percent' 'balancer-v1']	0xee9a6009...	0x2260fac5...	['percent' 'balancer-v1']
0xee9a6009...	0xc02aaa39...	['percent' 'balancer-v1']	0xf083fba9...	0x4e3fbd56...	['curve' 'bent']
0xf083fba9...	0x9e0441e0...	['curve' 'bent']	0xf35a80e4...	0xc02aaa39...	['degenerative' 'uma']
0xf861483f...	0x853d955a...	['fpi' 'curve']	0xf8ef02c1...	0xc02aaa39...	['degenerative' 'uma']
0xfcfc434e...	0x1498bd57...	['delta' 'core']	NaN	NaN	NaN

Table 11. **Duplicated balanceOf functions**. BalanceOf calls executed by different protocols on the same token address (column 'On') and input address (column 'Input').

A.5 TVL reconstruction from on-chain data

We provide additional details on the analyses conducted in Section A.5, starting from the procedures to obtain token prices. As the price time series extracted are noisy, we exclude the tokens having less than 10000 points, i.e. those that are traded rarely and have very low liquidity, assuming that they do not play a relevant role in the ecosystem. Next, we implement a cleaning procedure to remove from each price time series the outliers, i.e., individual price values that diverge by several orders of magnitude from the trend. To remove them, we compute the centered rolling average

on a time window of ten days and remove all values where the difference between the rolling average and the price time series is larger than 20%. While the approach is relatively simple, it provides satisfying results for most tokens, but it is not effective on tokens with low liquidity. We then conduct manual sanity checks to ensure that prices are cleaned correctly and additionally remove the prices of 80 tokens whose values have exceedingly high volatility and therefore it was not possible to determine accurately their correct trend. Future work could investigate these choices more thoroughly and provide an alternative scenario that includes also the removed tokens. However, as for most protocols the value is concentrated in few and widely used tokens, we do not expect findings to change substantially. Finally, we manually check that the stablecoins included in the study did not deviate from their peg and assume that their price is anchored to the US dollar.

Next, we discuss some additional limitations that may explain divergences in between vTVL and published DeFiLlama estimations. First, we noticed that for some protocols the API data are reported only after a certain date, while we find on-chain assets associated with them also before that date. This inflates the quantity of on-chain assets in comparison with the off-chain ones for earlier dates. Second, our infrastructure did not capture balance calls enabling to reconstruct the TVL of the protocol Lido, which is one of the largest protocols in terms of TVL according to DeFiLlama, especially after the end of 2022. Third, by investigating the largest protocols, we found that the API files do not always report data for protocol versions separated consistently with respect to how they are reported in the GitHub DeFillama repositories. For instance, Aave APIs report together Aave v2 and v3 (the repository reports one folder for Aave-v1 and one named 'Aave' seemingly for v2 and v3). The Uniswap API file reports together v2 and v3; however, Uniswap-v1, -v2, and -v3 are reported in a separate folder (we thus corrected API data accordingly). We also note that Uniswap -v1 and -v2 values are partly computed through the use of external hosts, while for Uniswap v3 we captured on-chain interactions. Similarly, a steadily discrepancy emerges for the protocol Maker at the end of 2022. Also in this case, as explained in the main body of the paper, this could be due to the use of external hosts to compute TVL, but this pattern is also consistent with the possibility that the APIs include both Maker and its related project Maker RWA. All these considerations explains at least in part some divergences that emerge in the comparison of on-chain and off-chain sources between and after end of 2022. Finally, as the change also corresponds with the Merge (which took place on September 2022), we cannot exclude that also this event has a role. This aspect needs further investigation.

A.6 Protocol categorization

Category	Included Protocol Types	N
DEXes	Dexes, DEX Aggregator, Cross Chain Swaps	79
Lending	Lending, CDP, NFT Lending, Uncollateralized Lending, RWA Lending	97
Derivatives	Derivatives, Options, Options Vault, Synthetics, Prediction Market, Leveraged Farming	47
Yield	Yield, Yield Aggregator, Farm	90
Staking	Liquid Staking, Liquid Restaking, Liquidity manager, Staking Pool	28
Other	Services, CEX, Bridge, Indexes, RWA, Launchpad, Insurance, NFT Marketplace, Reserve Currency, Gaming, Chain, Algo-Stables, Payments, Privacy, Wallets, SoFi, Oracle	173

Table 12. **DeFi protocol categories aggregated into high-level groups.** Note that not all protocols are labeled with a category.

Received 30 June 2026; revised XXXX; accepted XXXX