



Article

Abstracting Data in Distributed Ledger Systems for Higher Level Analytics and Visualizations

Leny Vinceslas ^{1,2}, Safak Dogan ^{1,*}, Srikumar Sundareshwar ³ and Ahmet M. Kondo^z ¹¹ Institute for Digital Technologies, Loughborough University London, London E20 3BS, UK² Institute of Sound Recording, University of Surrey, Guildford GU2 7XH, UK³ RegulAltion Ltd., Belmont Business Centre, Lewes BN8 6QL, UK

* Correspondence: s.dogan@lboro.ac.uk

Abstract: By design, distributed ledger technologies persist low-level data, which makes conducting complex business analysis of the recorded operations challenging. Existing blockchain visualization and analytics tools such as block explorers tend to rely on this low-level data and complex interfacing to provide an enriched level of analytics. The ability to derive richer analytics could be improved through the availability of a higher level abstraction of the data. This article proposes an abstraction layer architecture that enables the design of high-level analytics of distributed ledger systems and the decentralized applications that run on top. Based on the analysis of existing initiatives and identification of the relevant user requirements, this work aims to establish key insights and specifications to improve the auditability and intuitiveness of distributed ledger systems by leveraging the development of future user interfaces. To illustrate the benefits offered by the proposed abstraction layer architecture, a regulated sector use case is explored.

Keywords: distributed ledger technology (DLT); blockchain; block explorer; hyperledger fabric; abstraction layer; information visualization; analytics



Citation: Vinceslas, L.; Dogan, S.; Sundareshwar, S.; Kondo^z, A.M. Abstracting Data in Distributed Ledger Systems for Higher Level Analytics and Visualizations. *Future Internet* **2023**, *15*, 33. <https://doi.org/10.3390/fi15010033>

Academic Editors: Christoph Stach, Clémentine Gritti and Paolo Bellavista

Received: 1 November 2022

Revised: 6 January 2023

Accepted: 9 January 2023

Published: 11 January 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Distributed ledger technologies (DLTs) are becoming more widely used. They record operations between multiple parties in an immutable way. They are built on consensus-based decentralized systems, which address the trust issue between the involved parties. Numerous applications of distributed ledgers are currently being developed in various fields of the industry [1], such as agriculture [2], energy [3], finance [4], security [5], intellectual and digital property [6,7] or healthcare [8,9]. Figure 1a describes how a record of operations is maintained in a conventional centralized ledger. For example, a government or a bank may operate as a clearing house with complete control on the ledgers. In comparison, Figure 1b illustrates operation handling within a DLT framework, where each peer is maintaining its own ledger [10].

Motivated by the need to provide secure and decentralized services, DLTs keep track of very large amounts of ever-growing data [11]. By design, ledgers in DLTs persist low-level data that make conducting complex business analysis of the recorded operations challenging. Usually, the record of operations can be accessed by third party applications via querying the ledger [12,13]. This is achieved by employing a native set of application programming interfaces (APIs) where information about transactions, smart contracts or blocks can only be queried by their corresponding cryptographic hash. Although this access method allows for data lookup, such basic APIs are not adequate to devise high-level information from blockchains, often needed for analytics. Consequently, block explorers and similar visualization and analytics tools often only offer limited unintuitive information. These challenges call for an innovative approach for introducing a middleware between the presentation and data query layers that enables more accessible analytics and information

visualizations. This can be achieved by employing an abstraction layer, which aggregates data from the ledger, pre-processes it and provides higher level APIs to block explorers and analytics dashboards that can then intuitively present information readily.

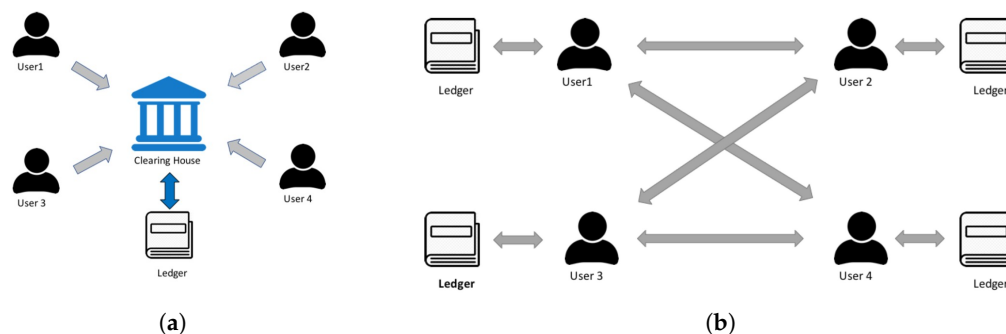


Figure 1. Transaction handling in different ledger architectures [10]. (a) Conventional centralized ledger. (b) Decentralized ledger.

In data analytics, visual representations are primarily designed to make sense of data and provide insights. They usually model data structures, which help to expand the boundary of individuals' cognitive system [14]. Not only do visual representations support users' reasoning, but also provide a construct to manipulate information. They can be used to both structure information and reduce individuals' cognitive burden by easing external anchoring, information foraging, and cognitive offloading [15].

In DLTs, audits are often conducted through lists and tables of low-level block data that do not easily allow for tracking and tracing digital assets [11]. The introduction of adequate visual representations of the ledgers' data has the capacity to enable a higher level of functionalities, and therefore improve the intuitiveness of the auditing processes.

A 2019 survey showed that time series representation accounted for 53% of all visualization techniques used for DLT representation, followed by basic charts such as bar charts, pie charts and histograms, which accounted for 41%, while tree and graph visualizations represented 38% [16]. The main reason behind this distribution can be explained by the time domain nature of the blockchain components, which contain time-stamped information. However, to represent data within more specific contexts, higher level types of visualizations were not used as frequently. For instance, map-based visualizations and other multi-dimensional representations accounted for only 13% and 7% of all visualization types, respectively.

To build intuitive visualizations or conduct an in-depth analysis of blockchain's recorded operations, data must be readily available in a format that can be consumed by the frontend visual representations. Although most DLTs provide access to the ledger via their software development kits (SDKs), they only offer low-level query interfaces that often lack semantic richness or functionality. For instance, information about transactions, contracts or blocks can usually be queried by their corresponding hashes. Such basic query interfaces are not adequate to map out high-level information or derive visualizations from blockchains' ledgers, and hence make it significantly challenging for users to deduce valuable insights that can serve complex business analysis quickly.

As a result, a few block explorers in the literature and marketplace implement high-level ledger visualizations or analysis functionalities. These limitations drastically reduce the possibility to adapt the abstraction of the displayed information to the targeted audience and application. The level of abstraction needs careful consideration when designing a visualization system to avoid situations where the users are presented with either insufficient or too much low-level data. In visualization and analytics, mismatched levels of granularity can result in both less accurate comprehension of a situation and higher time to complete a given task [17].

Curating information and presenting users with notable insights at the right level of granularity is often the responsibility of data analysts and designers. However, in the relatively new DLTs landscape, there are no such easily accessible solutions to implement the appropriate blockchain visualization tools to address a target audience's needs in a specific scenario [17].

For instance, firms of the regulated sectors willing to address compliance and trust issues might engage in decentralized digital assets trading by means of DLTs [18]. By design, DLTs encourage each participant to operate several peers on the network. For an effective and intuitive auditing process, visualization and analytics tools would need to represent the data at a participant-level, in an abstract and high-level manner. This requires aggregating the data from all the peers operated by a same participant. However, current off-the-shelf solutions typically provide information at a peer-level and are therefore limited to low-level representation of the ledgers' data [19].

The lack of high-level abstractions make the development of visual analytics time consuming for the developers since new functionalities need to be designed and implemented to analyze and aggregate the available low-level data. Additionally, delegating such functionalities to frontend applications can result in high resource-consuming services for the devices rendering such information. Therefore, there is a clear need for a standardized intermediate-level abstraction that provides higher level information from the low-level block data [19].

This article proposes an abstraction layer framework that facilitates better design of business analytics for DLT-based systems and the decentralized applications (DApps) that run on them. The purpose of this work is to establish a higher level of abstraction that improves the auditability and intuitiveness of distributed ledger records and enables further development of future user interfaces including analytics and visualization tools. Based on the analysis of existing initiatives and identification of the relevant user requirements, we infer specifications to improve the auditability and usability of block explorers. An abstraction layer has been designed to bridge the gap between a DLT and a user interface. As a result, the proposed abstraction layer coupled with a new user interface (UI) promotes the ease of analytics such as tracking and tracing of the history of operations, clustering of user addresses, and labeling of entities. Finally, to illustrate the benefits offered by the proposed abstraction layer architecture, we explore an industrial case study based on the RegNet platform [20]. RegNet is an infrastructure for trusted data access that removes the need to explicitly share data through the use of federated learning and tokenization. Its goal is to provide access to analytics in an auditable and privacy preserving manner, while being able to comply with data policies such as General Data Protection Regulation (GDPR) [21].

The remainder of this paper is organized as follows. We firstly review the related work by highlighting notable aspects in existing visualization tools, block explorers and abstraction layer implementations. Secondly, we propose an approach to build account and transaction-oriented abstractions while addressing auditability and intuitiveness issues. Thirdly, we provide an example of application in the RegNet case study. Finally, a discussion of the article is provided with an outlook on the future of the topic.

2. Related Work

In DLTs, the ledger is a global data structure collectively maintained by a set of mutually untrusting participants [22]. Changes to the ledger are organized into transactions which record the identifiers of their creators and beneficiaries. Transactions are hashed and grouped into blocks which are then chained together. Each block is appended via its header pointing to its predecessor. The synchronization of all peers on the state of the blockchain network is achieved using a consensus algorithm. This append-only ledger system provides DLTs with immutable records of transactions and therefore makes the blockchain tamper resistant. In addition to transactions, DLTs can implement smart contracts. Smart contracts

are executable scripts that read or write to the ledger and are deployed across peers of the network.

2.1. Visualization Tools

Visualization tools refer to pieces of software developed to represent DLTs' data through infographics. In contrast to block explorer, they do not usually provide extended search capability. These DLT's data representations can be sorted into different task domains [16]. Tools focusing on analyzing patterns of individual blockchain components, i.e., transactions, addresses and blocks can be classified under the transaction detail analysis task domain. For example, Blockchain Explorer proposes to visualize weekly or monthly transaction volumes as a tile map [23]. Ethviewer shows the real-time transaction pool in Ethereum using a node-link diagram to represent blocks and transactions [24]. BitExTract is a collection of visual analytic tools that analyses activities among Bitcoin exchanges, including transactional volume, market share, and connectivity between exchanges [25].

Tools representing information through a network and flows perspective can be classified in the transaction network analysis task domain. This category of representation is usually based on tree or node-link diagrams showing the connectivity among blockchain components. For instance, Daily-Blockchain provides a real-time representation of Bitcoin transactions where the nodes of the network evolve over time [26]. Bitforce5 only shows the most recent transactions [27]. This ensures a constant performance or rendering over time. For more granularity, BlockchainVis can either display the total amount of Bitcoin transactions or a specific address selected by the user [28]. Blockchain.com provides a tree diagram in which users can click through the tree levels to follow the value flow with respect to addresses [29]. Instead of presenting the value flow as a tree structure, BitConeView provides a unique diagram showing the value flow of a seed transaction as it appears in blocks from top to bottom [30]. Unlike the previous static value flow visualizations, BitInfoCharts dynamically represents the flow of transactions over the entire history of a blockchain utilizing a node-link diagram arranged in a linear layout [31].

Tools representing aggregated statistics of the network can be categorized under the network activity analysis task domain. For instance, Blockchain.com provides a long list of time series charts to display a wide range of Bitcoin network statistics, such as the total hash rate, average block size, total transaction fee, and mining difficulty [29]. BitNodes implements map-based visualizations where a node crawler gathers reachable Bitcoin nodes locations to estimate the global distribution [32].

The solutions proposed in the literature as well as by the online tools mainly focus on low-level aspects such as block creation, transactions and simple currency exchange taking place in distributed ledgers. These types of representations provide highly technical insights on network status, which do not facilitate intuitive audits of ledgers and might not be accessible for non-expert users. Moreover, due to their very specialised range of analysis, these tools do not offer comprehensive analytics of the data.

2.2. Block Explorers

Block explorers are tools that allow users to browse through ledgers including blocks, account addresses and transaction data. Block explorers are mostly search tools; however, they recently tend to adopt dashboard oriented layouts and integrate network activity analysis elements. This approach provides them with a more comprehensive design and improves their intuitiveness.

A variety of block explorers have been developed to analyze the transaction details and audit network activity of different distributed ledger platforms [33–37]. These are often based on lists and tables of data which fail to provide DApp-specific information easily. For example, Figure 2a shows the main view of the Hyperledger Explorer (HE) dashboard [33,38]. It presents users with statistical insights regarding the Hyperledger Fabric (HF) network (number of blocks, transactions, nodes and smart contracts) grouped by organizations or averaged over time. It also displays the name of the peers operating

on a specific channel and details about the last committed blocks. Figure 2b shows the block explorer view, which allows us to navigate into the network history and retrieve specific blocks. The transaction view displays the transaction history in a similar fashion than for the blocks. Figure 2c shows the transaction detail view which presents users with the details of a specific transaction when its ID number is selected.

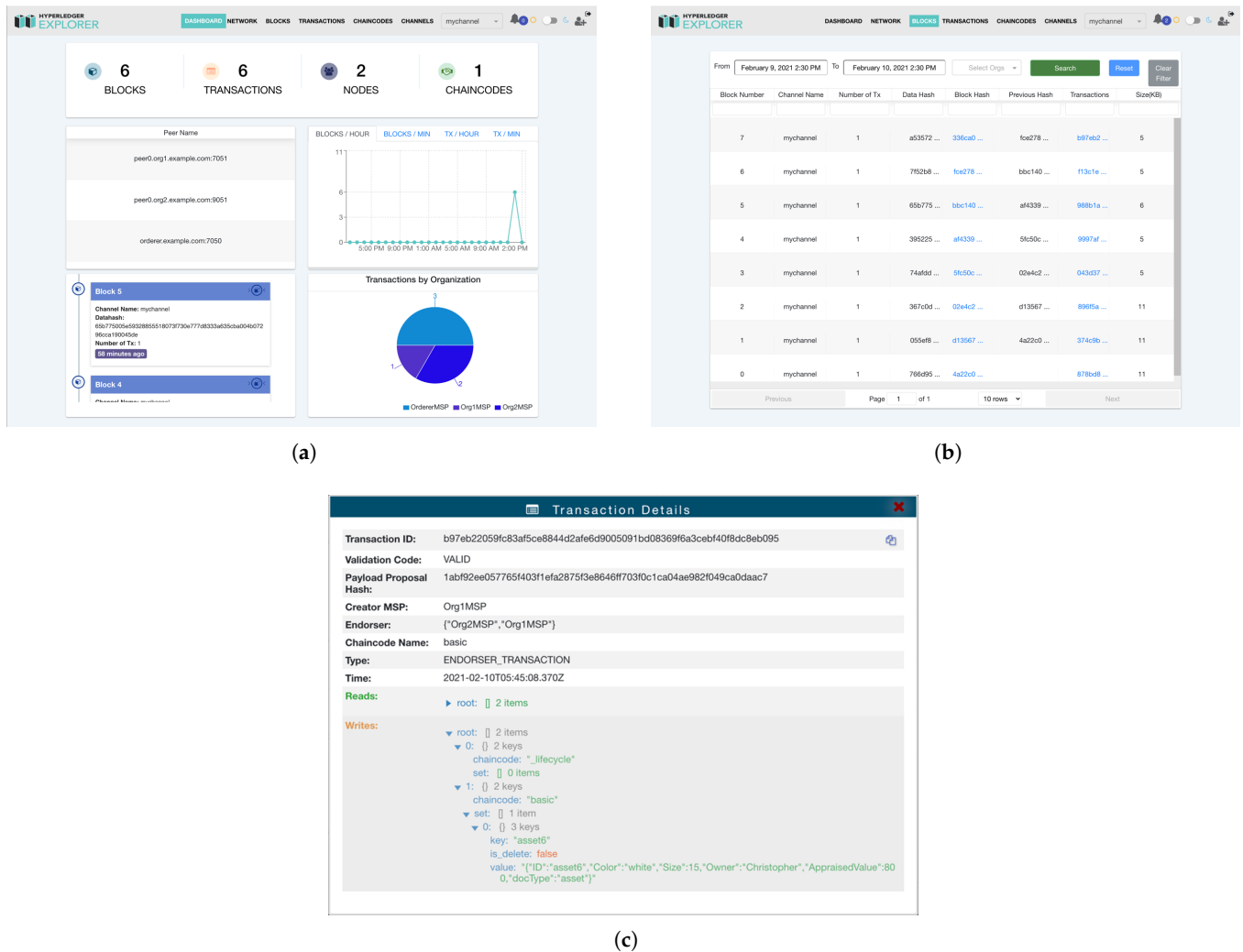


Figure 2. Screen-shots of Hyperledger Explorer [39]. (a) Dashboard main view; (b) block explorer view; and (c) transaction details pop-up window.

Alethio block explorer provides richer analytics [40]. In addition to the standard browsing history features, it maps interactions between accounts by tracing transactions and evoked smart contracts. These interactions are visually represented using simple node-link diagrams. It also allows us to keep track of account balances, search for information by account alias rather than by block and transaction hashes, and attach diverse social information to account addresses. Its functionalities, such as address tracking, tracing, labeling and data aggregation of DLT data improve auditability.

These block explorers appear to suffer from a lack of granularity in their presentation styles. They present users with very detailed information while failing at providing an overall context or general tendencies.

2.3. Abstraction Services

To overcome these limitations, both above mentioned block explorers implement an additional backend software sitting between the UI and ledger’s low-level query interface.

This standardized middleware aims to abstract the complexity of user interactions with blockchains, and is responsible for querying, aggregating, and conditioning the ledger data, so that it can offer higher level analytics more easily.

Several platforms have advocated for the need for abstraction layers. For instance, Ledgerdata Refiner is a ledger data query platform developed for interfacing permissioned DLTs such as HF [19]. It is based on a data analysis middleware, which extracts and synchronizes the ledger data, and then parses the relationship among them. From the queried blocks and transactions, the middleware provides end users with tailored queries to access aggregated ledger information.

Datachain is another example, which is an interoperable framework that eases the extraction of data from different underlying blockchains [41]. It allows users to define specific high-level query abstractions, and perform data requests, extract transactions, manage data assets and derive high-level analytic insights automatically.

More recently, The Graph proposed a decentralized indexing protocol for querying data off Ethereum and InterPlanetary File System (IPFS). In this framework, queries are based on the standardized API language GraphQL [42]. As previously introduced middleware, The Graph is also extracting, synchronizing, parsing and conditioning data from DLTs before returning it in a format that can readily be consumed in applications. The specificity of The Graph is that thanks to its decentralized architecture, it eliminates the trust issues between middleware services and client applications.

The difference between the aforementioned solutions and our requirements is two folds. Firstly, these block explorers and visual tools mostly display raw data from the ledgers with minimal contextualization. They seem to mainly target data analysts. Secondly, it would be counter-productive to employ cloud services for processing data from privacy-preserving DLTs. For these reasons we aim to develop a self-contained middleware that would enable the design of analytics and visualization with higher level of contextualization and accessible to general users.

3. Proposed Approach

We propose two types of visual representations: (i) a transaction-oriented abstraction emphasizing on the time series of the transaction history while allowing tracing and tracking of assets, and (ii) an account-oriented abstraction focusing on interactions between entities of the audited DLTs and providing insights on inter-party behaviors.

The transaction-oriented abstraction uses a directed acyclic graph layout. As shown in Figure 3a, vertices represent transactions while directed edges illustrate the transaction flow between the source outputs and target inputs. This visualization shows the flow of transactions relative to a given transaction. It allows tracking and tracing of assets from their origins to end points across a predefined number of hops. For a low granularity level, only one-hop tracking is displayed with respect to Tx₃, which corresponds to the blue vertices in Figure 3a. For a higher granularity level, a two-hop tracking is represented with blue and grey vertices. Using an adaptive design that displays details on demand, this visualization gives access to a continuum of granularity. In addition to the vertices, the directed edges can be augmented with asset values and transaction timestamps. Informing about the smart contracts that triggered the represented transactions can also provide pertinent insights. The implementation of this transaction-oriented abstraction requires knowledge about the mapping between the transactions of interest. However, this information is not directly available in the ledger and must be obtained through data parsing, aggregation and analysis.

The account-oriented abstraction uses force-directed graphs where different granularity levels are implemented. As shown in Figure 3b, when a macro level is chosen, the square-shaped vertices represent addresses while directed edges are illustrating interactions between accounts. At a lower level, circular vertices denote clusters of accounts, forming entities linked by the directed edges. Entities and directed edges can be of variable sizes, representing the quantity of accounts by cluster and total value or amount of all

inter-cluster interactions that occurred during a predefined time period, respectively. By nesting accounts within different cluster sizes, the visualization can efficiently adapt to the required level of detail. The implementation of the account-oriented abstraction requires knowledge on entities and their interactions, which needs clustering and labeling the ledger data.

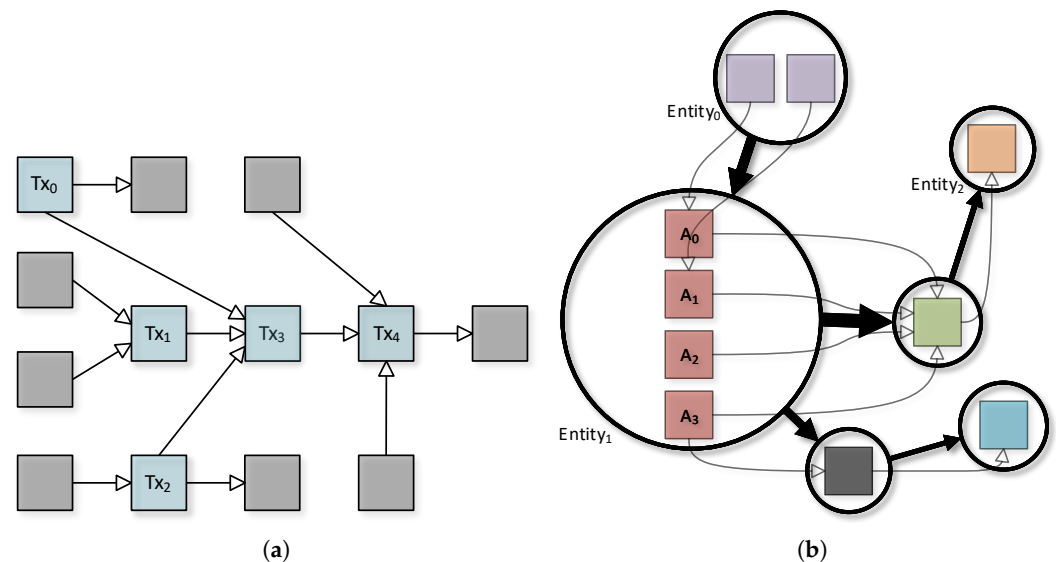


Figure 3. Abstracted visualizations. (a) Transaction-oriented abstraction; and (b) account-oriented abstraction.

To address the implementation of the two visual representations, we propose a general architecture for building an abstraction layer. The main purpose of this layer is to hide the complex interactions with ledgers. Through a simplified interface, users can connect to underlying ledgers to derive high-level analytic insights or perform high-level requests such as asset tracking and tracing. With this abstraction layer, we establish a framework where different services can be easily integrated to provide a transparent and richer query interface for business analytics. As depicted in Figure 4, it first extracts transaction, block, contract and channel details through blockchain ledger SDKs. At this stage, it is conceivable to query data from different DLTs. The ledger data is then parsed and aggregated to create comprehensive objects with common data structures, easy to manipulate in a given framework. The parsed data is then cached so that both current and historic data can be accessed by the pre-processing services. Pre-processing services aim at deriving high-level information from the low-level cached data. For instance, this is where information is mapped or filtered according to predefined heuristics. The last service of the abstraction layer provides interfaces to query the computed analytics.

- Parsing is the process of converting raw low-level data structures into higher level objects. Blockchain data structures are optimized for transaction validations and data retrieval across a distributed network and thus are not best suited for conducting analysis easily. For instance, to implement the proposed transaction-oriented abstraction, the parsing procedure must first collect the transactions from one or several blocks prior to mapping their inputs to previous transaction outputs. In addition, transactions must be assigned with IDs and timestamps along with the associated addresses of creators and beneficiaries to facilitate retrieval procedures [43] different parsing procedure;
- Aggregation refers to the collection and integration of data from multiple sources into a single storage destination. During this process, the different data sources required to infer higher level information are gathered and stored within a common data structure. For example, the proposed transaction-oriented abstraction needs to establish links between transaction inputs and outputs. To derive this mapping, transaction metadata

of different blocks is aggregated, and corresponding source and destination addresses are matched;

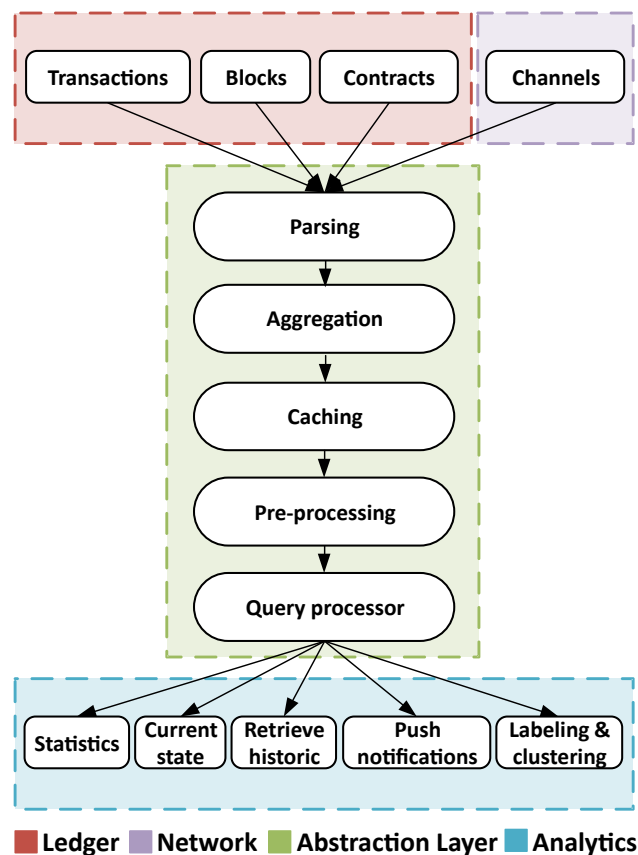


Figure 4. Abstraction layer architecture.

- Caching is the process of storing data resulting from previous computations so that future requests for that data can be executed faster. Both hardware and software used for caching depend on critical requirements such as the data volume, persistence time, access rate, throughput, and format. In the present scenario, the parsed and aggregated data can be cached in a server hard drive and RAM using a regular or graph database. The latter usually provides a better basis for analyzing relationships between entities [44];
- Pre-processing defines the operation of taking the cached data as input to generate the information requested by the query services. For example, this step is necessary to compute statistical insights on the network state, i.e., number of transactions per day. In addition to cached data, the pre-processing operation can also request data from third-party services. In the case of the proposed account-oriented abstraction, a pre-processing service will access the stored aggregated data to cluster addresses based on various possible heuristics [45]. Entities can then be inferred from the clustered accounts. Address clustering is particularly powerful when combined with labeling, i.e., labeling clusters with real-world entity designations [46]. On a small scale, labels can be determined by users through the query services. However, for large-scale labeling, automated scraping of open-source information or access to a third-party service provider is desirable;
- Query processor refers to the interfaces allowing third-party applications or users to query high-level data through a set of predefined instructions. Queries can initiate reading pieces of information collected or generated by the other abstraction layer services. Through a set of rich queries, this service aims to deliver requested data in a readily consumable format. To build the proposed visualizations, the query

services can be implemented using Representational State Transfer (REST) APIs and the JavaScript Object Notation (JSON) file format. The defined set of APIs will allow client applications to remotely execute pre-processing services to submit labels and clustering rules before querying the pre-processed data.

4. Use Case Scenario

4.1. The RegNet Platform

RegNet is a privacy-preserving data-access and data-collaboration platform for the regulated sectors, and addresses data-privacy challenges by combining DLTs, cryptography and machine learning [20]. Participants can request and provide access to each other's data while the resulting sharing agreements are stored in a distributed ledger. RegNet seeks to provide a trusted infrastructure to enable the exchange of data in a way where no sensitive information leaves the data-holders' firewalls. To ensure both security and relevance of exchanged information, RegNet uses privacy enhancing techniques on the data accessed between participants. In addition, RegNet also implements federated learning capabilities [47] that promotes a secure collaborative way to build larger data models together with semi-trusted participants.

RegNet's decentralized capabilities are provided by HF. HF is a permissioned distributed ledger platform targeting enterprise-grade business applications [48–50]. In addition to usual blockchain features, such as a decentralized ledger and tamper-proof data sharing, HF offers a more efficient consensus mechanism with higher throughput [51]. HF also addresses scalability and privacy issues by establishing the concept of channels. Channels allow the chosen data to be shared only among permissioned participants and thus provide a more adaptive data protection structure [52]. Areas of application include a digital vaccine passport [53], anti-counterfeit system [54], privacy-preserving in healthcare [55] and E-Voting System [56].

Since RegNet's decentralized capabilities are based on HF, HE appears as the default endpoint to access the ledger data and provide participants with monitoring and auditing features. However, the API implemented by HE is not entirely complying with the RESTful standard and does not allow the implementation of a suitable auditing layer for the data access applications that run on RegNet. In addition, a clear need for abstraction from HE's data arose. We therefore designed and implemented the discussed abstraction layer to more accessibly provide the interfacing and aggregating functionalities needed for analytics.

4.2. Architecture

The abstraction layer is designed and implemented as a middleware, which sits between the front-end dashboard application and the HE, as shown in Figure 5. The primary role of the abstraction layer is to hide the bad endpoints of the HE API from the dashboard API. The secondary purpose of this layer is to translate requests and responses so that the dashboard API can be compliant with the RESTful standards. This allows the developed applications to gain robustness for deployment while increasing their compatibility for future developments. In addition, this middleware provides the foundations for additional features such as persistence layers, data aggregation and processing, and authentication methods.

The abstraction layer was designed following a microservice architecture [57] using the Micronaut framework [58]. The microservice approach structures an application as a collection of smaller and consistent services. Under this type of architecture, microservices are separated autonomous components of an application, each accountable for a specific functionality and able to communicate with each other to form a coherent entity. The advantages of microservices oriented development are that it provides better maintainability in complex and large systems by enabling the deployment of many independent services, each of which may have a granular and autonomous life-cycle. An additional benefit is that microservices can scale out independently. Instead of having a single monolithic application that must be scaled as a unit, it can alternatively scale specific microservices to the

consumer application need and to an extent to the demand on the network. In our network visualization application, each microservice is deployed using Docker container images.

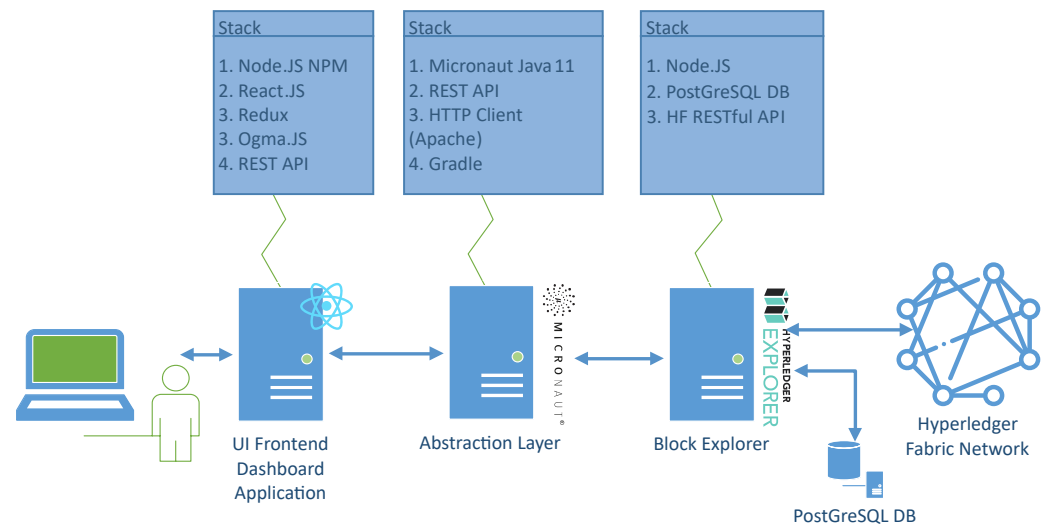


Figure 5. Application architecture and stack structures.

4.3. Consumer-Driven Contract Testing

When implementing microservice architectures, integration points between services can be a source of failure. Consumer-driven contract testing is an approach where the consumer of a service defines a contract and verifications are made against this contract within the provider’s test life-cycle [59,60]. Contract testing is also practical to test microservices in isolation before deploying them in a live environment. Depending on the scope and perspective of the testing, there are a number of tools available that can be used to implement contract tests [61,62]. To test integration points between our microservices, we employed the tool Pact [63]. Pact is a set of open-source libraries and frameworks for automating contract-driven testing, and is specifically well-suited for internal provider- and consumer-focused testing [64].

4.4. Queries

In the proposed design, the abstraction layer receives requests from the front-end application. If the requested data are not available in the persistence layer, the middleware queries the HE and external sources for the appropriate data. It is noteworthy that a single query from the front-end often results in a collection of different requests emanating from the middleware. After parsing, aggregating, processing and caching the data, the middleware responds back to the dashboard application providing the requested data in a JSON format.

4.5. Network Visualization

Figure 6 shows the implementation of one such account-oriented dataset access based analytics utility onto the RegNet platform. The visualization is rendered using Ogma, a JavaScript library for interactive graph visualization [65]. The implemented account-oriented graph features nodes that represent organizations and edges that specify the relationships between nodes. Nodes are clustered accounts belonging to the same organization and thus sum up all account activities of a participant. The number of data models made available by each organization is reflected by the node sizes while the node color is indicating the channel on which an organization operates. In HF, channels are separated ledgers that enable the privacy and the scalability of the platform. The width of the directed edges illustrates the quantity of access permission granted between two organizations, which is also numerically displayed. Adaptive granularity is introduced by

a tooltip providing a summary and further insights onto an organization when its node is double-clicked.

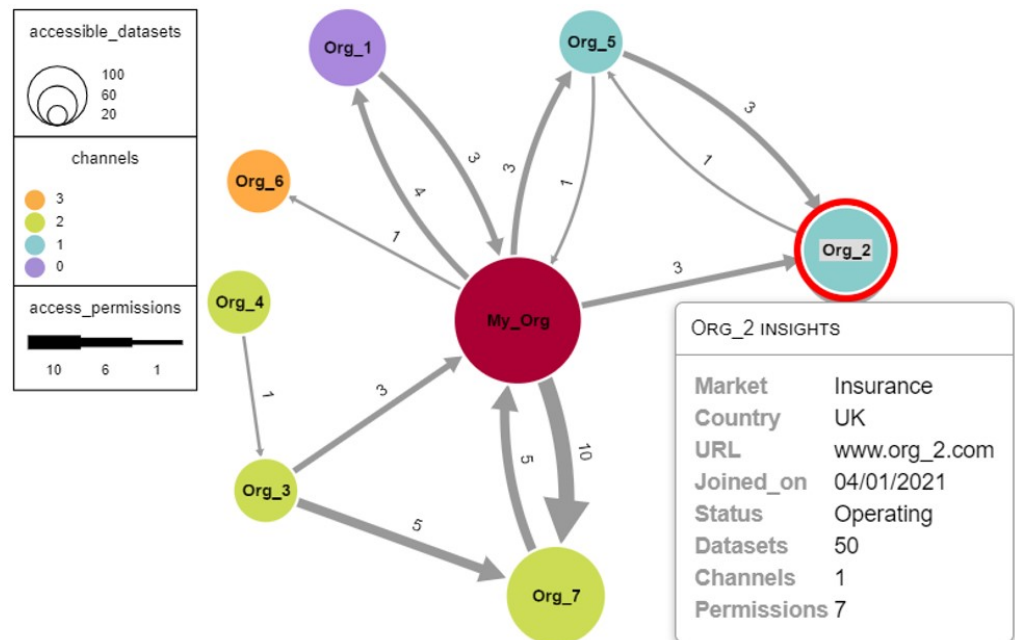


Figure 6. RegNet network visualization. The information used to populate the visualization is based on the data of the HF network deployed as a proof-of-concept in the RegNet case scenario.

4.6. Dashboard Concept

Figure 7 is a mock-up that proposes an integration of the network visualization in a dashboard framework. The design proposition is articulated around two main aspects: a news feed and a node-link diagram. The news feed presents users with the latest events that occurred in the network. New events appear at the top of the column, pushing the oldest ones toward the bottom in real time. Events can be filtered depending on their nature or grouped together for better visibility. The purpose of this presentation design is to highlight the time dimension of the network occurrences and enhance the perception of the dynamic aspect of the RegNet platform.

In this dashboard concept, users are able to interact back and forth with the news feed and the node-link diagram. For instance, when a specific event in the news feed is selected, the corresponding network components are highlighted. Moreover, a dynamic visual transition shows the direction or location of the corresponding action and its relation to other organizations or smart contracts. An additional window on the right side of the dashboard also displays the previous transactions between the highlighted entities and a plot of the number of transactions per day.

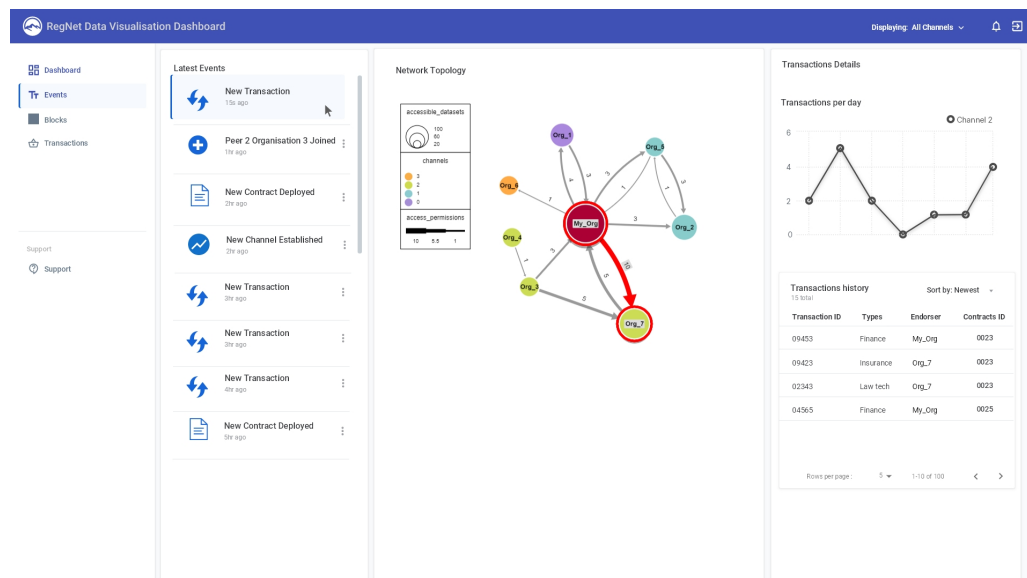


Figure 7. Concept design of the network topology view of the RegNet dashboard. From the left to the right column: page selection, news feed, network topology visualization and transaction insights. The analytics used to populate the dashboard are based on the data of the HF network deployed as a proof-of-concept in the RegNet case scenario.

5. Discussion and Outlook

In data analytics, visual representations are important since they can provide constructs that intuitively assist with inferring new information and can also reduce individuals’ cognitive burden. The introduction of adequate visual representations for ledger data enables a higher level of analytics and therefore augments the intuitiveness of auditing processes. Nevertheless, when designing analytics solutions, the level of abstraction needs careful consideration; higher level data can facilitate richer and quicker analytics. To apply these concepts, we designed two distributed ledger visual representations, i.e., transaction-oriented abstraction highlighting transactions history and structure while allowing tracing and tracking of assets, and an account-oriented abstraction focusing on interactions between entities and providing insights on inter-participant behaviors. To enable the implementation of these high-level visual representations, we proposed an abstraction layer architecture. Its main purpose is to provide coherent interfacing and aggregating functionalities allowing the production of readily consumable data. A comparison between the proposed approach and available abstraction services for HF is provided in Table 1. To illustrate the proposed visual concepts and application architecture, a use case based on a dashboard for the regulated sectors has been explored. Employing higher level abstractions to represent HF data enables better comprehension of entities’ interactions and improves the auditability of the system in comparison to HE.

As a result of their simplicity, the middleware and microservice architecture enable better maintainability and scalability of the system. The autonomous life-cycles of microservices allow them to be deployed at a relatively fast pace. However, due to their technical heterogeneity, a larger set of skills is required for their development and maintenance. In addition, to comply with the best practices, individual testing of the microservices needs to be performed. Both the necessary skill-set and the additional testing make this type of architecture less cost effective in the short term or at low scale compared to traditional monolithic applications. Yet, as a result of the high scalability and maintainability of the microservice architecture, this extra-cost can be recovered when developing and operating larger systems.

Ultimately, a universal higher level query language could be designed on top of this abstraction layer, which sits on blockchains. In turn, just like what Structured Query Language (SQL) is to Relational Database Management System (RDBMS), such language

with its compositional, pragmatic and rich semantics would make business level querying much easier.

Table 1. Comparative table of available abstraction services for Hyperledger Fabric and the proposed approach.

Categories	Features	Proposed Approach	Hyperledger Explorer [38]	Ledgerdata Refiner [19]	Datachain [41]
Architecture	RESTful API	✓			
	Microservice based	✓			
Data management	Processed data persistence	✓		✓	✓
	Ledger parsing & aggregation	✓		✓	✓
	Aggregation of external data	✓			
Low level queries	Block browsing	✓	✓	✓	✓
	Transaction browsing	✓	✓	✓	✓
	Block & transaction search by ID	✓		✓	✓
High level queries	Statistics on transactions	✓	✓	✓	
	Tracking & tracing transactions	✓			
	Ledger operation chronology	✓		✓	
	Network change report	✓			
Enabled Visualizations	Transaction flow & volume	✓			
	Organization & node activity	✓			
	News feed	✓			

Future work will investigate the scalability and interoperability of such system. In addition, a user study could also be conducted to evaluate the usability and effectiveness of the proposed visualizations and dashboard concept.

Author Contributions: Conceptualization, L.V. and S.D.; methodology, L.V. and S.D.; software, L.V.; validation, L.V., S.D. and S.S.; investigation, L.V.; resources, S.D. and A.M.K.; writing, L.V.; review and editing, S.D.; visualization, L.V.; supervision, S.D. and S.S.; project administration, S.D.; funding acquisition, S.D. and A.M.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Innovate UK [application number 45079; project number 106159].

Data Availability Statement: Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflict of interest.

References

- Casino, F.; Dasaklis, T.K.; Patsakis, C. A systematic literature review of blockchain-based applications: Current status, classification and open issues. *Telemat. Inform.* **2019**, *36*, 55–81. [[CrossRef](#)]
- Feng, H.; Wang, X.; Duan, Y.; Zhang, J.; Zhang, X. Applying blockchain technology to improve agri-food traceability: A review of development methods, benefits and challenges. *J. Clean. Prod.* **2020**, *260*, 121031. [[CrossRef](#)]
- Alladi, T.; Chamola, V.; Rodrigues, J.J.; Kozlov, S.A. Blockchain in smart grids: A review on different use cases. *Sensors* **2019**, *19*, 4862. [[CrossRef](#)]
- Ali, O.; Ally, M.; Clutterbuck; Dwivedi, Y. The state of play of blockchain technology in the financial services sector: A systematic literature review. *Int. J. Inf. Manag.* **2020**, *54*, 102199. [[CrossRef](#)]
- Kshetri, N. Blockchain's roles in strengthening cybersecurity and protecting privacy. *Telecommun. Policy* **2017**, *41*, 1027–1038. [[CrossRef](#)]
- Wang, J.; Wang, S.; Guo, J.; Du, Y.; Cheng, S.; Li, X. A summary of research on blockchain in the field of intellectual property. *Procedia Comput. Sci.* **2019**, *147*, 191–197. [[CrossRef](#)]

7. Wang, Y.C.; Chen, C.L.; Deng, Y.Y. Authorization mechanism based on blockchain technology for protecting museum-digital property rights. *Appl. Sci.* **2021**, *11*, 1085. [CrossRef]
8. Agbo, C.C.; Mahmoud, Q.H.; Eklund, J.M. Blockchain technology in healthcare: A systematic review. *Healthcare* **2019**, *7*, 56. [CrossRef]
9. Ratta, P.; Kaur, A.; Sharma, S.; Shabaz, M.; Dhiman, G. Application of blockchain and internet of things in healthcare and medical sector: Applications, challenges, and future perspectives. *J. Food Qual.* **2021**, *2021*, 7608296. [CrossRef]
10. Kadam, S. Review of distributed ledgers: The technological advances behind cryptocurrency. In Proceedings of the International Conference Advances in Computer Technology and Management (ICACTM), Pune, India, 23–24 February 2018.
11. Dinh, T.T.A.; Liu, R.; Zhang, M.; Chen, G.; Ooi, B.C.; Wang, J. Untangling blockchain: A data processing view of blockchain systems. *IEEE Trans. Knowl. Data Eng.* **2018**, *30*, 1366–1385. [CrossRef]
12. Zhu, Y.; Zhang, Z.; Jin, C.; Zhou, A.; Qin, G.; Yang, Y. Towards rich Qery blockchain database. In Proceedings of the 29th ACM International Conference on Information & Knowledge Management, Virtual, 19–23 October 2020; pp. 3497–3500.
13. Przytarski, D.; Stach, C.; Gritti, C.; Mitschang, B. Query Processing in Blockchain Systems: Current State and Future Challenges. *Future Internet* **2021**, *14*, 1. [CrossRef]
14. Hegarty, M. Diagrams in the mind and in the world: Relations between internal and external visualizations. In Proceedings of the International Conference on Theory and Application of Diagrams, Cambridge, UK, 22–24 March 2004; pp. 1–13.
15. Liu, Z.; Stasko, J. Mental models, visual reasoning and interaction in information visualization: A top-down perspective. *IEEE Trans. Vis. Comput. Graph.* **2010**, *16*, 999–1008. [PubMed]
16. Tovanich, N.; Heulot, N.; Fekete, J.D.; Isenberg, P. Visualization of blockchain data: A systematic review. *IEEE Trans. Vis. Comput. Graph.* **2019**, *27*, 3135–3152. [CrossRef] [PubMed]
17. Oscar, N.; Mejía, S.; Metoyer, R.; Hooker, K. Towards personalized visualization: Information granularity, situation, and personality. In Proceedings of the 2017 Conference on Designing Interactive Systems, Edinburgh, UK, 10–14 June 2017; pp. 811–819.
18. Polyviou, A.; Velanas, P.; Soldatos, J. Blockchain technology: Financial sector applications beyond cryptocurrencies. *Multidiscip. Digit. Publ. Inst. Proc.* **2019**, *28*, 7.
19. Zhou, E.; Sun, H.; Pi, B.; Sun, J.; Yamashita, K.; Nomura, Y. Ledgerdata Refiner: A Powerful Ledger Data Query Platform for Hyperledger Fabric. In Proceedings of the 2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS), Granada, Spain, 22–25 October 2019; pp. 433–440.
20. Treleaven, P.; Sfeir-Tait, S. *Future Data-Driven Regulation*; Technical report; UCL: London, UK, 2020.
21. Pithadia, H.J. Algorithmic Regulation using AI and Blockchain Technology. Ph.D. Thesis, UCL (University College London), London, UK, 2021.
22. Rauchs, M.; Glidden, A.; Gordon, B.; Pieters, G.C.; Recanatini, M.; Rostand, F.; Vagneur, K.; Zhang, B.Z. *Distributed Ledger Technology Systems: A Conceptual Framework*; Cambridge Center for Alternative Finance, Judge Business School: Cambridge, UK, 2018.
23. Kuzuno, H.; Karam, C. Blockchain explorer: An analytical process and investigation environment for bitcoin. In Proceedings of the 2017 APWG Symposium on Electronic Crime Research (eCrime), Phoenix, AZ, USA, 25–27 April 2017; pp. 9–16.
24. Ethviewer. Available online: <http://ethviewer.live> (accessed on 14 December 2022).
25. Yue, X.; Shu, X.; Zhu, X.; Du, X.; Yu, Z.; Papadopoulos, D.; Liu, S. Bitextract: Interactive visualization for extracting bitcoin exchange intelligence. *IEEE Trans. Vis. Comput. Graph.* **2018**, *25*, 162–171. [CrossRef]
26. Daily blockchain. Available online: <https://dailyblockchain.github.io/> (accessed on 14 December 2022).
27. Bitforce5. Available online: <https://www.bitforce5.com/> (accessed on 14 December 2022).
28. Bistarelli, S.; Santini, F. Go with the-bitcoin-flow, with visual analytics. In Proceedings of the 12th International Conference on Availability, Reliability and Security, Reggio Calabria, Italy, 29 August–1 September 2017; pp. 1–6.
29. Blockchain.com. Available online: <https://www.blockchain.com/explorer/> (accessed on 12 December 2022).
30. Di Battista, G.; Di Donato, V.; Patrignani, M.; Pizzonia, M.; Roselli, V.; Tamassia, R. Bitcoveview: Visualization of flows in the bitcoin transaction graph. In Proceedings of the 2015 IEEE Symposium on Visualization for Cyber Security (VizSec), Chicago, IL, USA, 25–26 October 2015; pp. 1–8.
31. BitInfoCharts. Available online: <https://bitinfocharts.com/bitcoin/explorer/> (accessed on 14 December 2022).
32. Bitnodes. Available online: <https://bitnodes.io/> (accessed on 14 December 2022).
33. Hyperledger Explorer Github. Available online: <https://github.com/hyperledger/blockchain-explorer> (accessed on 30 November 2022).
34. Etherchain. Available online: <https://etherchain.org> (accessed on 30 November 2022).
35. Ethplorer. Available online: <https://ethplorer.io> (accessed on 30 November 2022).
36. Etherscan. Available online: <https://etherscan.io/> (accessed on 30 November 2022).
37. Blockscout. Available online: <https://blockscout.com> (accessed on 30 November 2022).
38. Hyperledger Explorer Documentation. Available online: <https://blockchain-explorer.readthedocs.io/en/main/> (accessed on 14 December 2022).
39. Thinkit.co.jp. Available online: <https://thinkit.co.jp/article/18190> (accessed on 13 December 2022).
40. Alethio. Available online: <https://explorer.aleth.io> (accessed on 30 November 2022).

41. Trihinas, D. Interoperable Data Extraction and Analytics Queries over Blockchains. In *Transactions on Large-Scale Data-and Knowledge-Centered Systems XLV*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 1–26.
42. Tal, Y.; Jannis, P.; Brandon, R. The Graph. Available online: <https://thegraph.com/> (accessed on 30 November 2022).
43. Kalodner, H.; Möser, M.; Lee, K.; Goldfeder, S.; Plattner, M.; Chator, A.; Narayanan, A. Blocksci: Design and applications of a blockchain analysis platform. In Proceedings of the 29th {USENIX} Security Symposium ({USENIX} Security 20), online, 12–14 August 2020; pp. 2721–2738.
44. Tsoulas, K.; Palaiokrassas, G.; Fragkos, G.; Litke, A.; Varvarigou, T.A. A Graph Model Based Blockchain Implementation for Increasing Performance and Security in Decentralized Ledger Systems. *IEEE Access* **2020**, *8*, 130952–130965. [[CrossRef](#)]
45. Fröwis, M.; Gottschalk, T.; Haslhofer, B.; Rückert, C.; Pesch, P. Safeguarding the evidential value of forensic cryptocurrency investigations. *Forensic Sci. Int. Digit. Investig.* **2020**, *33*, 200902. [[CrossRef](#)]
46. Harrigan, M.; Fretter, C. The unreasonable effectiveness of address clustering. In Proceedings of the 2016 Intl UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld. IEEE, Toulouse, France, 18–21 July 2016; pp. 368–373.
47. Yang, Q.; Liu, Y.; Chen, T.; Tong, Y. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol. (TIST)* **2019**, *10*, 1–19. [[CrossRef](#)]
48. Androulaki, E.; Barger, A.; Bortnikov, V.; Cachin, C.; Christidis, K.; De Caro, A.; Enyeart, D.; Ferris, C.; Laventman, G.; Manevich, Y.; et al. Hyperledger fabric: A distributed operating system for permissioned blockchains. In Proceedings of the Thirteenth EuroSys Conference, Porto, Portugal, 23–26 April 2018; pp. 1–15.
49. Hyperledger Fabric Github. Available online: <https://github.com/hyperledger/fabric> (accessed on 30 November 2022).
50. Hyperledger Fabric Documentation. Available online: <https://hyperledger-fabric.readthedocs.io/en/latest/whatis.html> (accessed on 14 December 2022).
51. Nasir, Q.; Qasse, I.A.; Abu Talib, M.; Nassif, A.B. Performance analysis of hyperledger fabric platforms. *Secur. Commun. Netw.* **2018**, *2018*. [[CrossRef](#)]
52. Valenta, M.; Sandner, P. Comparison of ethereum, hyperledger fabric and corda. *Frankf. Sch. Blockchain Cent.* **2017**, *8*, 1–8.
53. Shih, D.H.; Shih, P.L.; Wu, T.W.; Liang, S.H.; Shih, M.H. An International Federal Hyperledger Fabric Verification Framework for Digital COVID-19 Vaccine Passport. *Healthcare* **2022**, *10*, 1950. [[CrossRef](#)] [[PubMed](#)]
54. Chen, C.L.; Shang, X.; Tsauro, W.J.; Weng, W.; Deng, Y.Y.; Wu, C.M.; Cui, J. An Anti-Counterfeit and Traceable Management System for Brand Clothing with Hyperledger Fabric Framework. *Symmetry* **2021**, *13*, 2048. [[CrossRef](#)]
55. Stamatellis, C.; Papadopoulos, P.; Pitropakis, N.; Katsikas, S.; Buchanan, W.J. A privacy-preserving healthcare framework using hyperledger fabric. *Sensors* **2020**, *20*, 6587. [[CrossRef](#)] [[PubMed](#)]
56. Diaz-Santiso, J.; Fraga-Lamas, P. E-Voting System Using Hyperledger Fabric Blockchain and Smart Contracts. *Eng. Proc.* **2021**, *7*, 11.
57. Microservices. Available online: <https://microservices.io> (accessed on 30 November 2022).
58. Micronaut. Available online: <https://micronaut.io/> (accessed on 30 November 2022).
59. Meyer, B. Applying ‘design by contract’. *Computer* **1992**, *25*, 40–51. [[CrossRef](#)]
60. Lehvä, J.; Mäkitalo, N.; Mikkonen, T. Consumer-driven contract tests for microservices: A case study. In Proceedings of the International Conference on Product-Focused Software Process Improvement, Barcelona, Spain, 27–29 November 2019; pp. 497–512.
61. Sotomayor, J.P.; Allala, S.C.; Alt, P.; Phillips, J.; King, T.M.; Clarke, P.J. Comparison of runtime testing tools for microservices. In Proceedings of the 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), Milwaukee, WI, USA, 15–19 July 2019; Volume 2, pp. 356–361.
62. Sotomayor, J.P.; Allala, S.C.; Santiago, D.; King, T.M.; Clarke, P.J. Comparison of open-source runtime testing tools for microservices. *Softw. Qual. J.* **2022**, 1–33. [[CrossRef](#)]
63. Pact. Available online: <https://docs.pact.io/> (accessed on 12 December 2022).
64. Ma, S.P.; Fan, C.Y.; Chuang, Y.; Lee, W.T.; Lee, S.J.; Hsueh, N.L. Using service dependency graph to analyze and test microservices. In Proceedings of the 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), Tokyo, Japan, 23–27 July 2018; Volume 2, pp. 81–86.
65. Ogma-Linkurious. Available online: <https://ogma.linkurious.com/overview> (accessed on 30 November 2022).

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.