LinkChains: Exploring the space of decentralised trustworthy Linked Data

Allan Third and John Domingue

Knowledge Media Institute, Open University, Milton Keynes, MK7 6AA, UK
{allan.third,john.domingue}
@open.ac.uk

Abstract. Distributed ledger platforms based on blockchains provide a fully distributed form of data storage which can guarantee data integrity. Certain use cases, such as medical applications, can benefit from guarantees that the results of arbitrary queries against a Linked Data set faithfully represent its contents as originally published, without tampering or data corruption. We describe potential approaches to the storage and querying of Linked Data with varying degrees of decentralisation and guarantees of integrity, using distributed ledgers, and discuss their *a priori* differences in performance, storage limitations and reliability, setting out a programme for future empirical research.

1 Introduction

This paper presents an exploration of the space of approaches to the storage and querying of Linked Data using distributed ledgers based on blockchains, in order to provide distributed data storage and query processing while maintaining guarantees of data integrity against corruption or tampering. These approaches can be useful in domains such as medicine, engineering and scientific data publishing, where data integrity can be of critical importance, and where distributed access to data would provide benefits in terms of availability and reliability.

Decentralisation has benefits other than increasing reliability by distribution. There is also an issue of control and trust. If important data is kept centralised, there remains the possibility of it being modified to suit the agenda of the central data controller. This is particularly a risk if the data is of financial importance to multiple, potentially disagreeing parties, particularly if the person or organisation controlling data also stands to profit from it. There can be political issues too, for example with climate data (see, e.g., [7]) or transparency in pharmaceutical trials. Decentralisation provides insurance against problems in these types of scenario, particularly if there can be strong guarantees of data integrity.

Since the introduction of Bitcoin [9], there have been significant efforts to extend its underlying *blockchain* technology to more general uses cases beyond simply cryptocurrency. The general notion of blockchains as secure, tamper-proof, fully-distributed append-only record stores has given rise to the term "Distributed Ledger" (DL). With the advent of blockchain platforms such as

Ethereum [16] capable of executing code, there has been (occasionally hyperbolic) talk of blockchains replacing the Web – the Ethereum Javascript library is even called `web3` – with part of the motivation being that blockchains are fully decentralised. Each node in a network has a full copy of the blockchain and has the potential to write to it; no node is privileged over any other.

More likely is that distributed ledger technology and the Web will develop to complement each other, where relevant (see, for example, [8]). This paper sets out the ways in which distributed ledgers can provide for the *integrity* of Linked Data sets and Linked Data queries, and how this varies with the degree to which data storage and querying are decentralised.

We present multiple approaches to this problem, differing in their levels of decentralisation and strength of guarantee of integrity offered, and compare them with each other. We begin by discussing Linked Data and distributed ledgers, and then present the different approaches available (potential "LinkChains") for the storage and querying of Linked Data, tabulating how they compare along various axes and finally discussing what both distributed ledgers and the Semantic Web are lacking in order to support the full benefits of a Distributed Semantic Web.

2 Linked Data

Linked Data is one of the key components of the Semantic Web – the notion of a large-scale machine-comprehensible Web of Data sitting alongside and complementing the existing human-readable Web. Linked Data is often associated with the Open Data movement, and there has been considerable success in encouraging the publication of large volumes of Linked Open Data for public reuse – see, e.g, [4]. The usual standard for Linked Data is the Resource Description Framework (RDF) [15], in which data are represented as triples – semantically, "subject predicate object" sentences – or quads – "graph subject predicate object" sentences – where graphs serve to group triples.

The aim of Linked Data is to support a Web of Data, and so, like the human-readable Web, is designed around the idea that anyone can publish data without any centralisation, and importantly, that once published, data can be *consumed* by anyone who knows where to find it. The idea is therefore to support more straightforward data integration, thanks to the common and simple data model, based on multiple authors of data and common querying approaches.

Issues with centralised Linked Data publishing include availability and mutability. A particular dataset is only available to be queried provided that its hosting datastore remains available online. Even if its contents are mirrored, if it contains URIs pointing to a particular server, those terms will cease to be dereferencable if that server no longer exists. The contents of a dataset may also be changed at any time, with no inherent means for clients to determine when this has happened. Alterations to data can occur for a variety of reasons: everyday updates by data authors, corruption or, potentially, deliberate data tampering or removal (for example, [7]). For many use cases, of course this mutability is an advantage, if it is important always to have the most up-to-date

data. For others, however, the opposite is true; it can be important to know that data has not been modified, or that it corresponds to a particular version of the dataset, for example, for retrospective clinical accountability.

3 Distributed Ledgers and Blockchains

A distributed ledger is an ordered list of records whose full contents are shared by nodes across a network, to which multiple authors can write in an append-only way. A distributed ledger is intended to be immutable – once a record has been appended, it effectively cannot be deleted or edited – and decentralised – there is no central authority with control over access to it. The most well known example of a distributed ledger is the Bitcoin infrastructure [9]; users are prevented from spending the same Bitcoins twice by reference to a distributed ledger recording every transaction in which Bitcoins are transferred.

These properties of distributed ledgers are ensured by the use of *blockchains*. A blockchain is a data structure consisting of a linked list of blocks, originating with a genesis block, with each block containing a set of transaction records. Every node on a blockchain network has a complete copy of the entire chain, and nodes compete for the right to aggregate ("mine") new transactions into a block and append that block to the chain, which is awarded by consensus of the whole network. Successful nodes are rewarded in some way (e.g., with coins in an associated cryptocurrency). There are different ways to implement the competition mechanism, with the most common being "proof of work", where nodes must demonstrate the solution to a hard computational problem. The main requirements are that there be a cost to adding a new block, and that if different nodes disagree on the contents of the next block ("forking" the blockchain), there is an incentive to resolve forks quickly according to a consensus. Once a block has been added to a chain, anyone wishing to rewrite its contents must convince the network as a whole to agree; the further back along the chain a block is, the harder and more expensive it is to do. Provided a blockchain network is sufficiently diverse (no more than 50% of all nodes owned or controlled by a malicious owner), the contents of a blockchain are secure from malicious alterations, and blocks a sufficient distance back along the chain from the most recent block may be regarded as effectively immutable (see [6]) and containing transactions which the network as a whole regards as having really happened.

Recent moves have taken distributed ledgers beyond storage of records to include distributed computation as well, with the idea of *smart contracts*. These are blobs of executable code stored on a blockchain with a published interface describing methods and their parameters. When a transaction which calls a smart contract method is mined onto the blockchain, that code is executed on all nodes of the network. Because the content of a smart contract is subject to the same promises of immutability as other kinds of blockchain data, and code and data can be signed cryptographically, smart contracts can be a form of trustworthy distributed computation. At the time of writing, the most developed smart-contract-based distributed ledger platform is Ethereum [16].

Of course it takes resources to run a smart contract, and, being written in Turing-complete languages, there is no way to guarantee that an arbitrary contract will terminate on arbitrary input. Invocation of smart contracts therefore involve a cost per significant step of computation.

Interfaces between distributed ledgers and the Semantic Web are in their infancy. FlexLedger [11] describes generic HTTP interfaces to blockchains, with a vocabulary implicit in the standardised names and responses of these interfaces. BLONDiE [14] and EthOn [10] formalise blockchain concepts as ontologies. [12] and [13] discuss initial approaches to RDF indexing of blockchains, and certification of RDF temporal streams on blockchains, but both are very preliminary.

4 Distributed File Storage

There are a number of options for distributed file storage with content-based addressing, and which can be used to provide some guarantee of file integrity, such as the Interplanetary Filesystem (IPFS) [5], Swarm [1] and FileCoin [2].

The essential idea of content-addressed distributed storage is that all nodes on a network share an index of files identified by the hash of their contents. When any client requests a particular hash, all nodes hosting all or part of that file respond and the contents can be copied by peer-to-peer filesharing to the requesting node, where the original file's contents are reassembled.

Files are immutable in the sense that any change to the contents results in a change of hash, making it impossible to access a modified file's contents using the original file's hash. However, there is generally no replication of data without a request, e.g., on IPFS, and it is possible for file content to disappear if the original host disappears and no cached copies remain on the network.

5 Data integrity and distribution

Distributed ledger technologies with smart contracts have the potential to provide immutable, trustworthy data storage and querying in a highly-decentralised manner. It should be noted that by "trustworthy", we mean "unmodified since initial publication and reliably timestamped". Of course if datasets are incorrect when first published, the usual "Garbage In, Garbage Out" principle applies.

We here present the range of broad approaches to the storage and querying of Linked Data in distributed contexts, and compare them according to: data integrity guarantee, distribution of data, distribution of query processing, cost of data storage and cost of data processing. In particular, we also consider whether data integrity for a particular query can be verified in time proportional to the size of the containing *dataset*, or proportional to the size of the query *results*.

6 Distributed trustworthy storage and querying

6.1 Base case

The default case for comparison is Linked Data stored in a single server in, for example, a triple or quad store, or embedded in HTML. Data may be distributed only effectively by duplication of the full datastore, with anyone wishing to use any copy of the data needing to know the precise location of the desired copy. Of course a Linked Dataset can be "distributed" across multiple datastores and be drawn together by queries – this, after all, is one of the main points of Linked Data – but in this case too, it is necessary to know the physical locations of each segment of the data, and it is anyway somewhat orthogonal to the point. For the sake of argument, let us assume that we are discussing a single dataset stored in one logical location, with linked integrations performed in the usual ways on top of the storage and querying described here.

Standard approaches, then, support only a manually-managed distribution of data. Distribution of query processing is not supported either: queries are evaluated in a single location, whether server or client. Storage and querying costs are ongoing and relate to the server costs of hosting. There are no guarantees of data integrity beyond what can be achieved by securing the relevant servers.

6.2 Content-addressed distributed storage (CADS)

There can be increased assurances of data integrity with content-addressed distributed storage, which could serve as filesystem in the backend of a Linked Data store, and, because CADS identifies a file by hashing its contents, these hashes can be returned with query results. A client can then, if desired, retrieve the file directly from CADS and verify its *contents*. In order further to ensure that dataset contents have not been modified, a client must also have a local copy of all relevant hashes to which to compare the results – otherwise a malicious user could replace both dataset contents and published hashes to ensure that tampered data still appeared genuine.

Here, querying is not distributed and query costs are higher than the base case, largely because verification of results requires copying a whole dataset. Data storage is distributed "on demand" - CADS files are mirrored across the network as needed. This has two consequences, which may cause a delay the first time a dataset is accessed, and lead to a risk of file contents becoming inaccessible if their original host deletes them or goes down and they are not cached on other nodes. Storage costs are therefore similar to the base case.

Using a distributed ledger with CADS (CADS+DL) A slight improvement to the data verification step of this approach could be made to avoid the requirement for clients to maintain secure lists of dataset hashes and timestamps, by placing, at the time of dataset publication, its hash and timestamp in a record on a distributed ledger. This would then permit checking of a hash and its timestamp in a trustable way without the need for a local copy.

The need to check the entire dataset from which results are drawn is expensive, potentially prohibitively so, as well as unnecessary and inelegant where query results may be very small in comparison with their source.

6.3 Distributed ledgers for storage and querying

Ideally, one could verify a query result in time proportional to the size of the query results themselves, and not the full dataset. We can do this by storing Linked Data directly on distributed ledgers. The following describes three approaches to doing so.

All approaches require a common data store, implemented as a contract. For consistent hashing, this requires a canonical string format for quads.

Base case with an accompanying distributed ledger (Base+DL) The standard query engine approach involves using a quad store as in the basic case, with the data kept in the store as normal. On retrieval of query results, these can be verified by transforming each quad in the result into the canonical format, and checking each individually using the methods of the smart contract datastore. This can be applied using a SPARQL server too, but requires some extra transformation of query results – SPARQL SELECT queries return variable bindings matching a given graph pattern; to generate quads, the graph pattern would need to be expanded with each variable binding in turn.

The benefits of this approach are that it provides query-level verification (as opposed to dataset-level) while maintaining a standardised infrastructure. Data is technically distributed in that it is duplicated across the distributed ledger by being in the smart contract datastore, but the copy of the datastore for querying is redundant. Query processing is similarly split, with query evaluation taking place in the standard datastore, and verification being distributed. While there is an extra cost in populating the smart contract datastore, the verification does not cost, as it involves only reading from a contract. The guarantee of data integrity inherits the guarantee provided by the distributed ledger, with the possibility to check data timestamps in a trustworthy way.

Base case with a distributed ledger backend (Base+DL backend) The redundant duplication of the data can be avoided by adapting a standard quad store to use the smart contract data store as a backend. Apart from the saving in data, this approach has similar properties to the preceding one.

"Pure" distributed ledger ("Pure" DL) For a fully decentralised solution, both data storage and querying can be carried out on a distributed ledger. To do so, we use the same smart contract datastore, and add a smart contract index and query engine. This method achieves fully distributed storage and querying of Linked Data on a distributed ledger with strong guarantees of data integrity. The trade-off is cost: as well as the initial cost of populating the smart contract data store, there is an execution cost for evaluating every query.

7 Comparison and Evaluation

Table 1 summarises the different approaches, with the aim of classifying of possible solutions which can be considered in the context of particular concrete use cases, and, importantly, to provide a programme for future empirical research.

	Base case	**CADS**	**CADS+DL**	**Base+DL**	**Base+DL backend**	**"Pure" DL**
Data distributed	No	On demand	On demand	Partial*	Yes	Yes
Queries distributed	No	No	No	No	No	Yes
Verification distributed	No	No	No	Yes	Yes	Yes
Ongoing data cost	Yes	Yes	Yes	Yes	Yes	No
One-off data cost	No	No	Small	Large	Large	Large
Query cost	Low	Medium*	Medium*	Low	Low	High
Integrity guarantee	None	Yes, cost for management	Yes, cost for verification	Yes	Yes	Yes

Table 1. Comparison of approaches to verifiable Linked Data storage and querying. Medium*: low query cost, high verification cost, Partial*: Yes, but a local copy of the data is also needed

We are in the process of implementing each of the non-base scenarios above in order to experiment with their real-world performance and cost in comparison with each other. However, doing so is only one step in the whole process of using Distributed Ledgers properly for the Semantic Web. There are a number of further issues to be solved; in particular, addressing on the distributed ledger side, and tools and standards for trust metadata on the Linked Data side.

Addressing is perhaps the most pressing of these. Fundamental to Linked Data is the notion that resources be addressable by URL. While Ethereum has the Ethereum Name Service [3], which allows human-readable names to be assigned to Ethereum resources, these names are not connected to standard DNS resolution. Individual nodes may of course have URLs and Web gateways, but nodes are potentially transitory and no one node is essential to a distributed ledger platform, by design. Ideally, akin to the "protocol, host, path" pattern for Web URLs, there would be a "chain type, chain, resource" URL format, where "chain type" specifies a particular distributed ledger protocol, (e.g., Ethereum or Bitcoin, "chain" specifies an individual instance of a ledger (e.g., the Ethereum main or developers' chain, the Bitcoin blockchain), resolvable via standard DNS, and "resource" identifies a particular entity or resource.

On the Linked Data side, the most common model for query results are the variable binding results provided by SPARQL endpoints, yet none of the specific

formats for this model, as far as we are aware, are able to carry metadata *about* the results. Ideally, it would be possible to insert trust and provenance metadata (as RDF) in a result set for clients to access easily.

8 Conclusion and Future Work

We have described and categorised multiple different approaches to the verifiable storage and querying of Linked Data on distributed ledgers, comparing them with each other along multiple axes.

We are in the early days of exploring the potential of distributed ledgers and their role in supporting architectures for verified claims. This potential will only be realised and fully exploited with standard means of connecting them to the existing data architecture of the Web, and the wealth of existing work on Linked Data and the Semantic Web.

References

1. (2017), http://swarm-gateways.net
2. (2017), http://filecoin.io
3. (July 2017), https://ens.domains
4. Abele, A., McCrae, J.P., Buitelaar, P., Jentzsch, A., Cyganiak, R.: Linking Open Data Cloud Diagram 2017. http://lod-cloud.net/ (2017)
5. Benet, J.: IPFS: content addressed versioned P2P fs. arXiv:1407.3561 (2014)
6. Buterin, V.: http://ethereum.stackexchange.com/a/203 (2016)
7. Eilperin, J.: Under Trump, inconvenient data is being sidelined (2017), https://www.washingtonpost.com/politics/under-trump-inconvenient-data-is-being-sidelined/2017/05/14/3ae22c28-3106-11e7-8674-437ddb6e813e_story.html
8. English, M., Auer, S., Domingue, J.: Blockchain technologies & the Semantic Web: A framework for symbiotic development. In: CS Conference for University of Bonn Students, J. Lehmann, H. Thakkar, L. Halilaj, and R. Asmat, Eds. pp. 47–61 (2016)
9. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)
10. Pfeffer, J., Beregszazi, A., Detrio, C., Junge, H., Chow, J., Oancea, M., Pietrzak, M., Khatchadourian, S., Bertolo, S.: EthOn - an Ethereum ontology. https://consensys.github.io/EthOn/EthOn_spec.html (2016)
11. Sporny, M., Longley, D.: Flex Ledger 1.0. W3C Blockchain CG (2016)
12. Third, A., Domingue, J.: Linked Data indexing of distributed ledgers. In: Proceedings of the 1st International Workshop on Linked Data and Distributed Ledgers at WWW 2017. pp. 1431–1436. WWW 2017 (2017)
13. Third, A., Tiddi, I., Bastianelli, E., Valentine, C., Domingue, J.: Towards the temporal streaming of graph data on distributed ledgers. In: 2nd International Workshop on Linked Data and Distributed Ledgers, Supplementary Proceedings of the 14th Extended Semantic Web Conference (forthcoming 2017)
14. Ugarte, H.: BLONDiE. https://github.com/EIS-Bonn/BLONDiE (2016)
15. W3C: Resource Description Framework. https://www.w3.org/RDF/ (2014)
16. Wood, G.: Ethereum: A secure decentralised generalised transaction ledger. Ethereum Project Yellow Paper (2014)