Decency and Decentralisation: Verifiable Decentralised Knowledge Graph Querying

Conference or Workshop Item

Decency and Decentralisation: Verifiable Decentralised Knowledge Graph Querying

Aisling Third and John Domingue
Knowledge Media Institute, The Open University, UK
{aisling.third,john.domingue}@open.ac.uk

Abstract. Increasing interest in decentralisation for data and processing on the Web brings with it the need to re-examine methods for verifying data and behaviour for scalable multi-party interactions. We consider factors relevant to verification of *querying* activity on knowledge graphs in a Trusted Decentralised Web, and set out ideas for future research in this area.

Keywords: Trust, Decentralisation, Knowledge Graphs, Federated Querying, Solid

1 Introduction

There is increasing public awareness that personal data is valuable and vulnerable, and that data about individual behaviour is routinely aggregated and exploited for commercial gain. It is widely recognised that centralisation is a factor in enabling this exploitation; large providers such as social networks integrate many services, of their own and from third parties, tying them to digital identities which they provide and control, allowing them to serve as hubs for data collection and analysis. This model requires that *trust* be placed in the centralising parties: trust that identity will be faithfully represented, personal data will be kept securely and privately, and data processing will only be carried out in line with stated policies. Generally, the only view an individual has on data processing is from seeing the results for them of using a centralised service; whether their data was copied or used for anything else is not transparent.

Technologies for a decentralised Web seek to provide standardised alternatives to this model which return agency to individuals. Linked Data Platforms such as Solid **[1]** are designed to support these models. Third-party Web applications for these platforms would no longer run against large databases of information aggregated across all users of a centralised site, but instead query each user's personal data store directly whenever needed.

The goal of this paper is to set out a high-level vision for ways that Web applications could demonstrate *decent* data behaviour when it comes to decentralised knowledge graphs and querying, with the hope of stimulating discussion and interest in future research and experimentation into the topic.

1.1 Scenario

We focus in particular on a scenario in which individuals' personal data is kept in private personally-controlled "pods" with common access and programming interfaces, prominently exemplified by the Solid Linked Data Platform infrastructure, and with a Web application landscape as described in [4]. We envisage a decentralised Web where Web apps operate on individually-controlled personal data stores, querying, with authorisation, for personal data at the time of need for processing.

Figure 1 illustrates a common centralised scenario for a Web application with multiple users, where the application provider hosts user data in their own infrastructure, shown as being within the application, and users communicate directly with it.The centralised provider controls access to, and what happens with, user data.

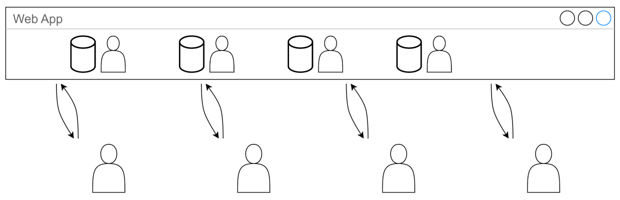

Figure 1. Centralised Web applications

Figure 2(a.) shows the user-centric view of an architecture of Web applications based on personal data stores, essentially as presented in [4]; individual users control their own data in "pods", and Web applications retrieve only *relevant* and *permitted* data from them in order to function for that individual. The motivation is that the individual is the source of truth and controller with regard to their own data rather than multiple potentially distinct providers in silos.

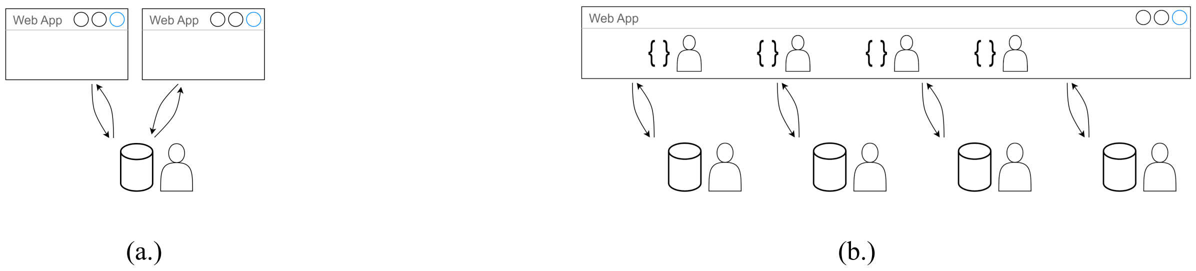

Figure 2. Decentralised Web applications based on personal data stores - (a.) user- and (b.) app-centric views

This only shows part of the picture, however, because only one user is shown. Considered from the perspective of a single Web application, however, Figure 2(b.) illustrates the likely scenario of it having multiple users. In contrast to Figure 1, user data is not *controlled* by the application as a matter of architecture, but some quantities of user data (shown as { }) are, at least transiently, *held* by it.[1] Bearing in mind that an extremely common requirement is that applications support *interactions* between users, there is an inevitable need in this case to link multiple users' data together, potentially in arbitrary large numbers. There is therefore a risk that many of the same issues arise as in the model in Figure 1. The two scenarios are not, of course, identical: data held by the user in 2(b.) which is *not* exposed to the application can be protected from misuse in a way which is not possible in 1, but for that data which is exposed, the risks are the same.

A key element of this landscape is that pods are *personal knowledge graphs*, and that applications interacting with pod data can use Linked Data standards to do so. Applications, then, are performing *federated knowledge graph querying* across the set of pods they are authorised to access.

2 Decency and Decentralisation

We can ask the question, then, of whether there are any technical methods which could be applied for application providers to guarantee well-behaved data processing to users, with the hypothesis that such decent behaviour could be a selling point in a competitive Web ecosystem. In principle, some elements of what follows could be applied meaningfully in other scenarios; however, we assume the use of personal data stores in order that data minimisation

[1] Assuming the application does not run entirely client-side, although a comparable situation could still be constructed with regard to one client interacting with other users' data.

be applicable at the data source, and more specifically the Solid ecosystem for its designed-in use of knowledge graphs.

Given the above, we consider the following features which could be offered to provide an audit trail for users of data decency:

1. *Traces*: records of inputs, outputs, and code execution logs
2. *Policy* validation: user data usage policies validated
3. *Verification*: tamper-evident data and known-good verifiable code
4. *Anonymity*: data exposure prevention measures

The expectation would be that whenever an application performs a query on an individual's pod, information based on these are sent to the pod as a querying receipt. In considering these, we are not looking for whether the outcomes are accessible for non-technical users to understand. While this is an essential requirement for any solutions to be *useful* at scale, it is a very large and multilayered topic in its own right and deserves separate thought.

2.1 Traces and Transparency

For someone to know what has been done with their data, some form of *trace* of querying activity is needed. Very generically and naively, this could include *input* and *output* data, and, for thoroughness, any intermediate outputs too. Shared with the user, such a trace could provide *transparency* of application activity.

2.2 Policy and Purpose

When it comes to controlling what happens with data, it would be beneficial if preferences for what should and should not be permitted could be represented in a machine-readable form. Several formalisms for this purpose exist (see, e.g., [2]), essentially offering ways to represent *constraints* on data based on arbitrary conditions, as well as representations of *purpose* for processing, in the form of *data usage policies*.

Suppose, then, that a resource in a data pod is associated with a policy describing how the individual permits that data to be used for which purposes. How can this be used to affect what applications can *do*? For this, we need a policy validation engine[2] which determines programmatically whether a given use of data complies with a given policy. It may do this, for example, as a data authorisation check – a pod may refuse to give access to data for non-compliant processing. But once data leaves the pod, we may wish the policy to be attached to it, in order for the application provider to generate evidence that they have behaved well, by validating traces of processing against the policy. Notice that conceptually here, the policy and records of (non-)compliance are simply input and output for a data processing step (policy validation) which would now be run by the application, and therefore can be traced and made transparent as above.

2.3 Aggregation and Anonymity

In general, the assumption that input data would come only from a single individual does not hold; we would typically expect applications to be processing data from multiple people, not in isolation, but linking them. A common example would be when two users interact on a social network. It is therefore very important to make sure that any tracing and policy validation infrastructure respects interpersonal privacy too. That is to say, a trace and policy validation proof for one user should not reveal confidential data for any other users. Otherwise, sharing traces with users is likely to constitute a privacy breach rather than prevent one. It may, of course, particularly when data

[2] For example, https://github.com/oeg-upm/odrl-lib

has been aggregated or inferred by a query, be difficult to identify or separate all and only the outputs of a query which relate to a single individual. However, with knowledge graph data, and access to a query engine's *query plan*, and coupled with metrics for dataset privacy and anonymity [5], it may be possible to take advantage of the available structure to track data from input to output, and perhaps even to quantify or determine limits on what input data can be reconstructed from the results of aggregation operators, etc., based on operator semantics.

2.4 Versions and Verifiability

As a basic guarantee of integrity, timestamped tamper-evident traces can enable a user to know when a trace was generated and whether it has been modified since that time. Noteworthy in this context are forms of tamper-evident data which support *selective disclosure*; that is to say, given a tamper-evident dataset, one can extract tamper-evident subsets of it. Selective disclosure allows partial traces (for example, relating to personal data of only one individual) to be shared without sacrificing verifiability, and can therefore enable privacy-preservation in multi-user situations as above. Approaches such as our own Linkchains MerQL [3] are examples of selective disclosure verifiability which are specialised for Linked Data. To be sure that a trace honestly represents an execution of well-behaved code, code verification methods could also be employed, whether via digital signatures of open-source or audited *versions* of packages or execution environments such as smart contracts. In the general case, this would be impractical with arbitrary application code due to commercial confidentiality or rapid development needs. However, in the scenario in question, the software components which need to be verifiably versioned are generic: a knowledge graph query engine and a policy validation engine. It is therefore potentially practical for a trace to refer (verifiably) to the particular versions of open-source components which were used to generate it.

4 Conclusion

We have set out a high-level vision consisting of a set of ideas which we argue are collectively relevant to providing trustworthy querying of decentralised knowledge graphs in a verifiable and privacy-preserving way. Given this, we propose the following research topics which we believe would be fruitful towards testing this vision: analysis of query operators and planning across multiple open-source query engines, including an analysis of operators for their data aggregation/deanonymisation effects, a model of how individual data items pass through query execution, an investigation of privacy boundaries for federated querying and verifiable selective disclosure, and the semantics and requirements for expressing and validating data usage policies over query plans and execution traces.

Acknowledgments. Text

References

1. E Mansour, AV Sambra, S Hawke, M Zereba, S Capadisli, A Ghanem, A Aboulnaga, and T Berners-Lee. 2016. *A Demonstration of the Solid Platform for Social Web Applications*. In Proceedings of the 25th International Conference Companion on World Wide Web (WWW '16 Companion). International World Wide Web Conferences, Switzerland, 223–226. https://doi.org/10.1145/2872518.2890529
2. S Steyskal and S. Kirrane. 2015. *If you can't enforce it, contract it: Enforceability in Policy-Driven (Linked) Data Markets*. International Conference on Semantic Systems (2015).
3. A Third, and J Domingue, (2019). *LinkChains: Trusted Personal Linked Data*. In: Blockchain-enabled Semantic Web at the International Semantic Web Conference, 27 Oct 2019, Auckland, New Zealand.
4. R Verborgh, *Paradigm Shifts for the Decentralized Web,* Retrieved: 2023-02-14 https://ruben.verborgh.org/blog/2017/12/20/paradigm-shifts-for-the-decentralized-web/,
5. I Wagner and D Eckhoff. 2018. *Technical Privacy Metrics: A Systematic Survey*. ACM Comput. Surv. 51, 3, Article 57 (May 2019), 38 pages. https://doi.org/10.1145/3168389