Ethics and Executability: Tracing Decency in Decentralised Knowledge Graph Applications

Aisling Third[1] and John Domingue[1]
[1] *Knowledge Media Institute, The Open University, 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 previously identified as relevant to verification of processing activity on knowledge graphs in a Trusted Decentralised Web, and use an implementation scenario to identify focused open questions.

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; it is not transparent whether their data was copied or used for anything else. Technologies for a decentralised Web seek to provide standardised alternatives to this model which return agency to individuals. Linked Data Platforms such as Solid **[4]** 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.

"Trust", in general, is a broad term with multiple facets. One might trust the outputs of a service if they repeatedly prove to be accurate or useful; perform well in terms of reliability or responsiveness; give value for money; behave ethically, and so on. We focus on trust that a service provider is *well-behaved* with personal data in the sense of, e.g., being ethical, doing only what the data's subject permits, being transparent and truthful about what has been done, and protecting the user's privacy in both processing and transparency. Ethical behaviour which is reassuring to users about service activity is what we call *decent*. In [8], we argued for a set of factors relevant to Web applications demonstrating *decent* data behaviour when it comes to federated querying of decentralised knowledge graphs, with a combination of data verification technologies, machine-readable data usage policy validation, and traces of data processing activities. Here, we very briefly summarise that paper, and step through a scenario in which those factors could be taken into account. From this, we identify topics which need to be explored to enable even a very simple scenario based on favourable assumptions to be handled.

2. Decency and Decentralisation

We are interested in a landscape in which individuals' personal data is kept in private personally-controlled "pods" with common access and programming interfaces, prominently exemplified by the

Proceedings of the ESWC 2023 Workshops and Tutorials, May 28–June 1, 2023, Hersonissos, Greece
EMAIL: aisling.third@open.ac.uk (A. 1); john.domingue@open.ac.uk (A. 2)
ORCID: 0000-0002-0386-1936 (A. 1); 0000-0001-8439-0293 (A. 2)

Solid Linked Data Platform infrastructure, and with the architecture described in [9] of a decentralised Web where Web applications operate on these pods, retrieving only *relevant* and *permitted* data from them in order to function for that individual, only querying, with authorisation, for that data at the time of need for processing. The motivation is that the individual is the source of truth and controller of their own data rather than multiple potentially distinct providers in silos.

Figure 1 shows two perspectives on this architecture: the user- and the app-centric views. In (a.), the focus is on the user, and their individual selection of Web applications which they want to have some access to their data. This only shows part of the picture, however, because only one user is shown. Considered from the perspective of a single Web application, however, (b.) illustrates the likely scenario of it having multiple users. A key element made visible by the perspective in (b.) is that pods are *personal knowledge graphs*, and therefore that applications interacting with multiple pods are performing a type of *federated knowledge graph querying* across the pods they are authorised to access.

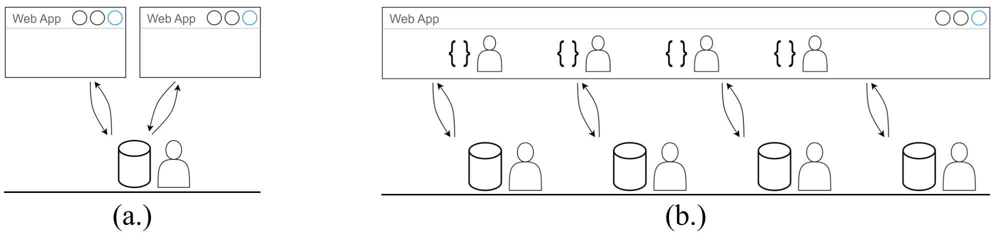

Figure 1 Decentralised Web applications based on personal data stores - (a.) user- and (b.) app-centric views

In [8], we argued for the following set of technical considerations which could be applied in order for application providers to guarantee well-behaved query processing to users, with the hypothesis that such decent behaviour could be a selling point in a competitive Web ecosystem.

1. *Traces*: records of activity, e.g., input, intermediate, and output data to promote *transparency*.
2. *Policy* validation: machine-readable constraints on data usage for different purposes, based on arbitrary constraints. Relies on policy validation engines to test a particular use of data against a policy. See, e.g., [6].
3. *Verification*: timestamped tamper-evident data in shareable formats, including verifiable selective disclosure (e.g., [7]). Known-good verifiable code.
4. *Anonymity*: interpersonal privacy respected both by platforms and by verifiable tracing & policy infrastructure - a trace and policy validation proof for one individual should not reveal personal information of another.

As well as policy evaluation and verifiable selective disclosure, this concept builds on existing work on provenance tracing and trusted execution. Recalling that we consider decency to include not just being well-behaved, but also being reassuring to users, it is important to look at how good behaviour may be implemented *and* how it can be communicated. In terms of federated querying of knowledge graphs, the implementation of tracing depends on each individual query, as data gathered from different sources is processed with various operations by a query engine to produce results. Operations which, for example, aggregate data can lead to individual results originating from multiple data sources, which potentially can be anonymising (if operations cannot be reversed algorithmically or by estimation), or transparent (if they can) - in the latter case, sharing a full trace with the subject of one data source may reveal information about the subject of another. Provenance tracing in the database querying literature (e.g., [1]) identifies subtypes of provenance: *lineage*, *why*-, and *how-provenance*. In brief, for a query result, its lineage is the set of data sources contributing to it, its why-provenance is the set of paths in the query plan/execution between the result and the sources in the lineage, and its how-provenance is (an algebraic representation of) the why-provenance with the query operators that are applied at each stage along the path. For knowledge graph querying, [3] presents a means of tracking provenance of queries by means of query rewriting, giving how-provenance (and therefore why-provenance and lineage) for non-aggregating queries, and lineage for aggregating queries - with the specific advantage of the query rewriting approach being its compatibility with arbitrary query engines. This is particularly relevant in the decentralised/federated context where different data sources may be accessed via different engines. At the time of writing, we could not identify any research into privacy/anonymity analysis of algebraic how-provenance for knowledge graph queries. For communicating provenance information, vocabularies such as the Provenance Ontology[2] and VoID[3] are well-established for describing activities and datasets; there is however a gap with regard to common vocabularies for describing how-provenance and query plans.

Trusted execution and technologies to support it cover a range of interpretations of trust: see, e.g., [5] for a thorough overview. Often, these technologies include dedicated hardware in some form, to minimise the possibility of malicious or misbehaving software causing an unfounded sense of trust. While commonly-used trusted execution hardware (e.g., contactless payment cards) might only support the execution of certain cryptographic operations on secure data, others can secure applications or virtual machines. As an alternative to dedicated hardware, blockchain-based smart contract solutions (e.g., the Ethereum[4] Virtual Machine) supports trusted execution of public code by means of highly-distributed public computation. In general, the security provided by blockchain technologies relies on multiple parties executing the same code on the same data, which inherently decreases privacy and performance without careful design to avoid risks. Recent work such as [2] explores trusted execution for knowledge graphs using smart contracts.

2.1. Scenario

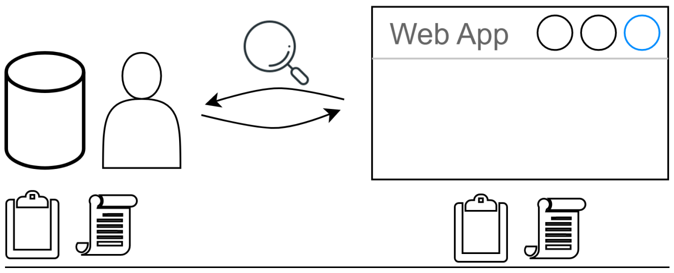

Figure 2 An application interacting with a pod

In order to understand the issues and difficulties, we walk through a scenario of an interaction that takes the preceding considerations into account. For clarity, consider Figure 2, a scenario with only two actors: a user, represented by their pod, and a Web application representing its provider. Both actors have their own policies on the use of their data, and both have activity tracing. We look at the case of the user seeking reassurance that the application is well-behaved with their data; in reality, of course, this can work both ways. The application is seeking to query the user’s data, in line with their policies, to produce output. We assume that this processing will also involve data from other users, and that the output may be sent to further actors which may or may not include the user shown. We also, for the sake of argument, make the significant simplifying assumptions that inputs and outputs are all Linked Data, and that application providers are using open-source components for at least data transformation and aggregation (if not, e.g., user interface). These are clearly highly unrealistic assumptions which hide difficulties that actual application of these ideas would encounter; however, even with these simplifications, there are many open questions to be addressed. Understanding the issues in the simple scenario will hopefully then shed light on how to approach more complex real situations.

[2] https://www.w3.org/TR/prov-o/
[3] https://www.w3.org/TR/void/
[4] https://ethereum.org/en/

Taking the proposal of our previous paper, we step through a hypothetical data query and processing scenario and consider key points at which our suggested use of activity tracing, policy evaluation, verification, and considerations of anonymity come into play. The specific mechanisms of verification are omitted: digital signatures, distributed ledgers, or other technologies could play this role. For reasons of space, we also omit most depictions of specific activity tracing - *every* action in this scenario is logged and traced by the relevant actors, and traces are anchored as tamper-evident. If absent, it should be assumed that it has taken place.

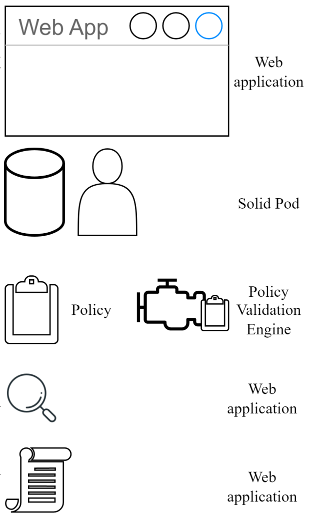

Figure 3 Key

Step 1. Code Verification

Beginning from a point at which the application and user have authenticated with each other, the user consents in principle to the application querying and processing some of their data. The user will rely on activity traces and policy evaluation from the application to determine behaviour, and therefore needs to know that the code used for these purposes meets their own policies and has not been modified to, e.g., create false records of activity. As we have assumed components are open source, the application can provide cryptographically-backed proof that the particular query and policy validation engines to be used (indicated in Figure 4 by the lock symbol) correspond to known-good public versions which can be subject to community scrutiny. If relevant, the user can do the same. **Questions:** how can we ensure the application uses the components it presents, and does not substitute or wrap them with something else that misbehaves? Can approaches to trusted execution be practically used here?

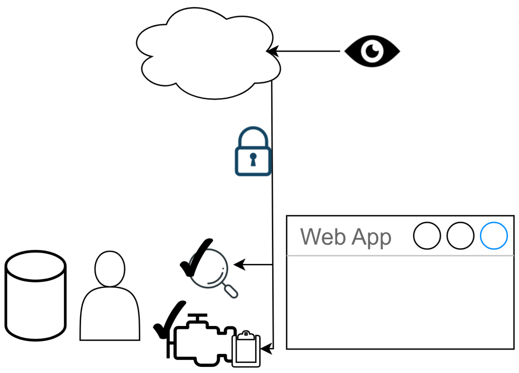

Figure 4 Code verification

Step 2. Policy-based access control

The application's verified query engine (represented by the magnifying glass in Figure 5 sends a query to the user's pod via the *user's* policy validation engine. This reads the user's data usage policy, and their pod, and validates that the query and its results comply. The results are only returned to the application if this succeeds. The data ({}) and a copy of the user's policy are then passed to the application by its query engine. **Questions:** What policy formalisms are suitable in terms of both expressiveness and automated validation? What vocabularies and granularities of the *purposes* of processing activities are useful, and how is compliance with purpose to be evaluated?

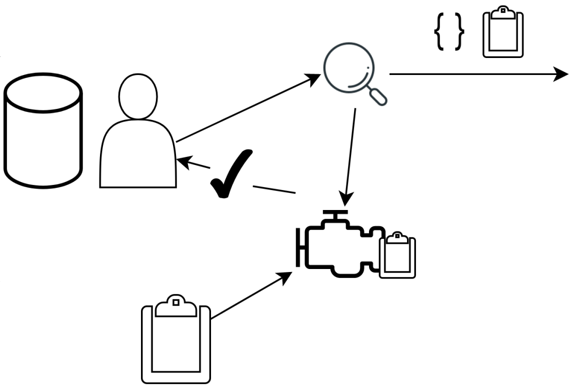

Figure 5 Policy-based access control

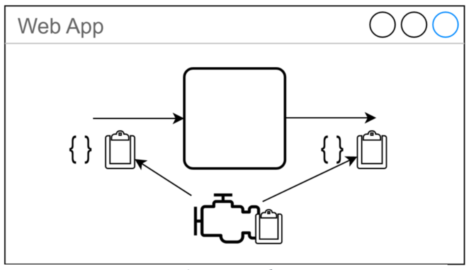

Figure 6 Internal processing

Step 3. Application processing

Internal to the application, the user's data is processed, transformed, and aggregated with other users' data to generate outputs. The *application's* verified policy validation engine has access to the user's policy, the (user's contribution to) input data, and the output data, as shown in Figure 6. It validates the output with respect to the policy. **Questions:** What are the relevant types of criterion to be expressed in policies that could allow for automated validation of compliance on outputs, and, for example, for anonymity, what are the useful metrics? What are the minimal sets of information which traces need to include for this? And what are the trade-offs and constraints with regard to user and others' privacy, and application confidentiality, which affect and limit what can be done here?

Step 4. Trace anonymisation
The preceding step is the only step which involves data originating from neither the user nor the application. Thus, while the application's activity traces of prior stages could potentially be shared with the user, for the processing stage, this might no longer be safe from a privacy perspective, as data might leak between users via the traces. The application therefore needs to ensure that the verifiable records of activity returned to one user cannot be used to extract anything which can be identifiably linked to another, and that anything which *could* enable that is sent only to the user it identifies. Figure 7 illustrates this. **Questions:** This step may even be unnecessary if verifiable automated policy evaluation can provide strong enough guarantees of compliance. In the (not unlikely) event that it cannot, what are the possibilities and limitations in trying to separate or selectively disclose trace contents?

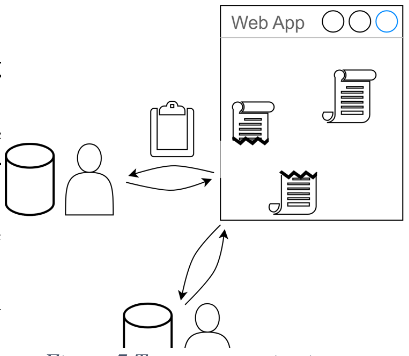

Figure 7 Trace anonymisation

3. Conclusion

We have described a scenario in which we attempt to bring a previously-identified set of technical factors to bear on the task of making decentralised Linked Data querying transparent and traceable. Specifically, we considered key stages in such a process and identified relevant open questions to be answered to enable even a very simple scenario based on favourable assumptions to be handled. As we progress ongoing work into infrastructure to implement these ideas, we are seeking to answer these questions, among others that will undoubtedly arise, with the goal of laying the groundwork to build on for realistic scenarios, to improve transparency and control over data usage for everyone.

4. References

[1] J Cheney, L Chiticariu and W-C Tan. 2009. *Provenance in Databases: Why, How, and Where*. Foundations and Trends in Databases: 1(4), pp 379-474. http://dx.doi.org/10.1561/1900000006.

[2] V Goretti. 2022. *Safe and controllable information consumption for data market applications*. Masters thesis, University of Rome La Sapienza, https://penni.wu.ac.at/supervision/Valerio%20Goretti%20Master%20Thesis%202022.pdf.

[3] D Hernández, L Galárraga, and K Hose. 2021. Computing how-provenance for SPARQL queries via query rewriting. Proc. VLDB Endow. 14(13), 3389–3401. https://doi.org/10.14778/3484224.3484235.

[4] 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). Switzerland, 223–226. https://doi.org/10.1145/2872518.2890529.

[5] C Shepherd, G Arfaoui, I Gurulian, RP Lee, K Markantonakis, RN Akram, D Sauveron, and E Conchon. 2016. *Secure and Trusted Execution: Past, Present, and Future - A Critical Review in the Context of the Internet of Things and Cyber-Physical Systems*. IEEE Trustcom/ BigDataSE/ ISPA, Tianjin, China, 2016, pp. 168-177, https://doi.org/10.1109/TrustCom.2016.0060.

[6] 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).

[7] A Third and J Domingue. 2019. *LinkChains: Trusted Personal Linked Data*. In: Blockchain-enabled Semantic Web, International Semantic Web Conference 2019, Auckland, New Zealand.

[8] A Third, and J Domingue. 2023. *Decency and Decentralisation: Verifiable Decentralised Knowledge Graph Querying*. Trusting Decentralised Knowledge Graphs and Web Data at the Web Conference. http://oro.open.ac.uk/88166/.

[9] R Verborgh. 2017. *Paradigm Shifts for the Decentralized Web,* Retrieved: 2023-02-14 https://ruben.verborgh.org/blog/2017/12/20/paradigm-shifts-for-the-decentralized-web/.