Towards Provable Provenance and Privacy-Preserving Queries in Decentralised Data Architectures

Jesse Wright[1]

[1]University of Oxford, Department of Computer Science, Wolfson Building, Parks Rd, Oxford OX1 3QG

Abstract

This research investigates how to add provenance to SPARQL queries, and support queries over distributed sources - whilst minimising unecessary information disclosure. To achieve this, we employ Zero Knowledge Proof (ZKP) to minimise provenance, whilst retaining integrity guarantees - and Secure Multi-Party Computation (S)MPC to reduce data-sharing when querying across distributed data-stores.

Keywords

Verifiable Credentials, SPARQL, Zero Knowledge, Zero Knowledge Proof, RDF, JSON-LD

1. Problem statement

This work develops a standardised declarative query language (data sublanguage) for accessing graph database(s) alongside zero-knowledge verifiable provenance statements – including of data sourcing, integrity and derivations. Supported queries include "Is Jesse over 21 according to facts issued by EU or UK governments" – the verifiable response reveals only the answer: "yes." This query language is first implemented in query engine(s) which evaluate queries over a locally indexed graph database. Support is then added for queries over the union of data residing across independent and potentially malicious graph-databases; by developing algorithms and architectures which minimize data disclosure between sources when planning and executing queries.

In present research the verifiable data sublanguage is a SPARQL 1.1 [1] interface with proof and provenance information added to results bindings. Next, we plan to develop a SPARQL 1.2 [2] interface with custom built-ins. Non-SPARQL interfaces may be investigated in the future.

Minimal Use Case

A group of four friends are applying for a rental. They each have a Solid Pod [3], containing W3C Verifiable Credentials [4], which include their last years' worth of payslips. They need to prove to their landlord that their cumulative salary is over £100k p.a. to rent a property.

With current Verifiable Credential standards [4], the landlord must fetch all the friends' payslip credentials directly. The landlord's application can then verify the payslips and calculate the aggregate salaries. The work to implement the data sublanguage over a local database allows the landlords application ask the applicants Pods "is this person's annual salary over £25k," and have a proven answer provided to the landlords application on demand. Support for queries over the union of data residing across independent databases allows the landlord to ask of the four Solid Pods "is the cumulative salary across these wallets greater than £100k."

ISWC 2025 Companion Volume, November 2–6, 2025, Nara, Japan

jesse.wright@cs.ox.ac.uk (J. Wright)
https://www.cs.ox.ac.uk/people/jesse.wright/ (J. Wright)
0000-0002-5771-988X (J. Wright)

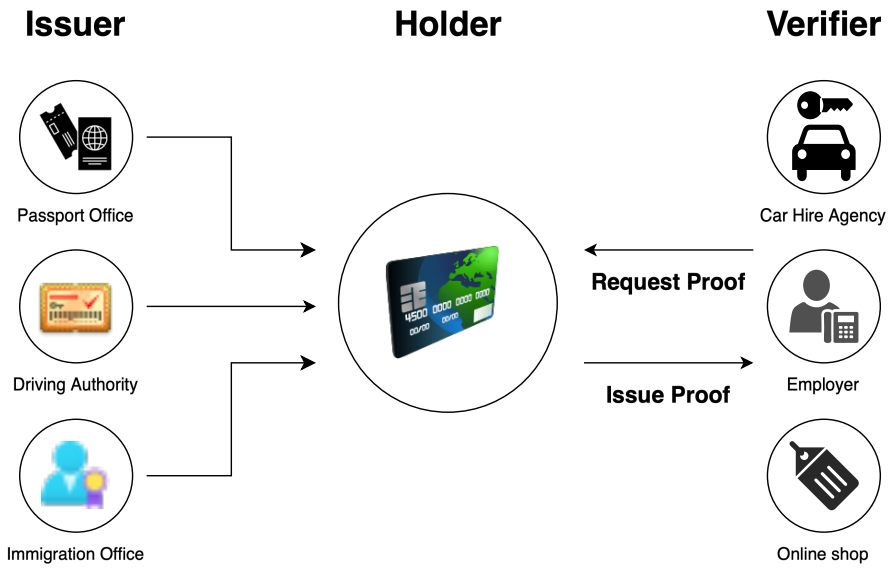

Figure 1: The typical relationship between issuers, holders and verifiers in Verifiable Credential flows. Solid Pods can act as holders.

2. Importance

Verifiable Credentials (VCs) [4] are seeing widespread adoption [5]. Driven by governments and regulation - many Australian states have had digital drivers licenses [6] since 2016; in Europe eIDAS (Electronic Identification, Authentication, and Trust Services) 2 regulation [7] is mandating that all EU citizens will have access to at least one European Digital Identity (EUDI) wallet [8]; and the UK's Data (Use and Access) Bill [9] is likely to bring in parallel regulation to the UK in the form of the Digital Verification Scheme (DVS). Verifiable Credentials are also becoming heavily relied upon for supply chain traceability. The United Nations Transperancy Protocol (UNTP) [10] - for instance - uses Verifiable Credentials to prevent conterfeiting, support sustainiability audits and improve automated decision making.

Verifiable Credentials refer to a suite of W3C (World Wide Web Consortium), ISO (International Standards Organisation) and IETF (Internet Engineering Task Force) **standards for signing data**. As shown in Figure 1, there are three entities typically defined within these standards: the **issuer** – responsible for creating and signing the data, the **holder** – who collects signed data from the issuer and is usually the data subject, and the **verifier** – who the holder forwards the signed data to as needed. In the Section 1 minimal example, the Solid Pod [3] is acting as a holder service. The format of the credentials - and interfaces for transporting credentials between *issuer*, *holder* and *verifier* varies greatly between the W3C, ISO and IETF standards.

There are two primary **ISO** standards for Verifiable Credentials: the Mobile driving license (mDL) specification [6] and the Cards and security devices for personal identification (ISO23220) specification [11]. The mDL defines a fixed schema of approximately 30 attributes (including name, address, and date of birth) for digital driver's license's that may be encoded as JSON or CBOR. ISO23220 extends this to other identity documents, such as passports, residency permits, and building passes. Both specifications fix the concepts that can be expressed to specific data models, and do not have a (formal logical) semantics for the data encoded.

The **IETF** is standardising JSON based Verifiable Credentials called SD-JWT-based Verifiable Credentials [12] - based on JSON Web Tokens (JWT's) [13] which are commonly used for authentication online. This standard does not restrict what can be expressed within credentials, but also does not support formal semantic data models out-of-the-box

The **W3C** Verifiable Credentials 2.0 Data Model [4] provides a general model for adding integrity proofs to JSON-LD [14] documents. This standard does not restrict what statements can be made

within the credential - and by requiring the use of JSON-LD - supports RDF [15] and formal semantics out-of-the-box.

ISO, W3C and IETF all support **Selective Disclosure (SD)** as a feature for preserving privacy when using Verifiable Credentials. SD enables a *holder* to reveal a subset of facts within a credential to a *verifier* and prove those facts are true without requiring the *issuer* to issue a new credential. SD is commonly advertised as enabling privacy preserving age verification with digital drivers licenses [16]. SD is typically implemented using the BBS signature scheme [17]. The holder of a BBS signature can generate zero-knowledge proofs of knowledge of the signature, while selectively revealing a subset of the signed messages. This signature scheme is derived from the 2004 work of Boneh et. al [18]. SD has limited expressivity to proof of set containment. To support proving that one is over 18 - as promised in mobile drivers licenses - issuers must sign the statement `is_over_18` [6] rather than holders deriving this statement from a date of birth.

As we discuss in Section 3, theory and tooling exists to support zero-knowledge proof of arbitrary computations. There is clear demand for more advanced privacy features than SD in Verifiable Credentials to support use-cases such as that presented in Section 1 and proof-of-age from birth date; evidenced by active work on this topic in the Credentials Community Group (CCG) [19] with which the author is actively engaged.

Observing that *verifiers* must be able to describe the information they require from *holders* - it is the authors view that the most sensible way to provide these privacy features is through a query interface. Further, the query language must: use globally unique identifiers [1] to support distributed data sources out of the box; be compatible with the Open World Assumption (OWA) [20] so that it is not possible to produce invalid results by choosing to omit credentials from the input; and have clear execution semantics [21] such that the expected results-set is well-defined whilst enabling query engine implementations to be specialised to the deployment environment – this is the case with most query languages such as SQL [22], SPARQL [1] and Cypher [23]. Crucially, the semantics of the query must not be dependent on the endpoint it is executed against, as is the case with query languages such as GraphQL [24]. The SPARQL Protocol and RDF Query Language (SPARQL) [1] supports these three requirements.

Ideally, the query language should capture provenance, including issuer signatures, natively within the database and query result. This could be supported with the use of SPARQL 1.2 [2] to discuss reified terms (that is, statements about statements). This, the author intends to support a SPARQL 1.2 interface with proof built-ins as part of this work.

3. Related Work

3.1. Zero Knowledge Cryptgraphy and zkSNARKS

Contemporary Zero Knowledge Cryptographic techniques support more expressive proofs than Selective Disclosure (SD). We outline a subset of techniques (abstractions) for creating Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (zkSNARK) [25]. zkSNARKs allow provers (holders) to generate a full proof without interacting with the verifier. SD proofs generated from BBS Signatures [18] are zkSNARKs.

Zero Knowledge Arithmetic Circuits builders (ZK Circuits) [26] - such as circom [26], plonky3 and Halo 2 [27] provide an abstraction for creating zkSNARKs. Circuit builders generate zkSNARKs proving whether a set of variables satisfies a given set of mathematical constraints. Which mathematical expressions are supported in the constraints is dependent on the circuit builder - with circom [26] supporting quadratic constraints. Gu et. al [28] prove that circuit builders can be used to generate zkSNARKs of correct execution of SQL queries with their implementation of PoneglyphDB [28] using the Halo 2 Circuit Compiler [27].

Zero Knowledge Virtual Machines (ZKVMs) enable proof of correct execution of a given instruction set. RISC Zero [29] is one such Zero Knowledge Virtual Machine which proves that a set of **outputs** have been generated by correctly executing a RISC-V instruction set [30] - without revealing

any information about the input or execution trace. Since higher level languages including Rust can be compiled into RISC-V instruction sets, it is possible to prove whether a set of outputs have come from the correct execution of any Rust code - provided there are no system calls. There numerous other ZKVMs including Ceno [31], SP1 [32], Nexus21, Powdr [33] and ZkMIPS [34].

zkSNARKs for RDF Datasets - Braun et al. [35] have developed a set of zkSNARK capabilities for RDF terms and datasets which do not depend on circuit or ZKVM abstractions. Specifically, they support proof of numeric bounds on terms, equality of terms between triples, set non-membership and selective disclosure at a term level. These proofs can be generated in subsecond speeds on consumer hardware. Whilst not directly shown in Braun et al. [35] - the capability of proving equality of terms between triples and selective disclosure at a term level implies the ability to prove that a BGP pattern is satisfiable by data across a set of credentials.

3.2. Distributed query evaluation with (secure) multi-party computation ((S)MPC)

We now present prior work on distributed query evaluation with (S)MPC. This includes a discussion of Domain Specific Languages (DSLs) for (S)MPC [36] and distributed database implementations that use (S)MPC.

There are more than a dozen (S)MPC DSLs. These include the Secure Multiparty Computation Language (SMCL) [37] which was developed as declarative programming language for Secure Multi-party Computations, but does not contain descriptions for the security assumptions required for MPC calculations. Wys* [38] - co-developed by Microsoft Research and the University of Maryland presents a DSL for (S)MPC which provides program logic to reason about the correctness and security of (S)MPC programs.

Most work on (S)MPC over distributed storage is targeted at relational databses, typically with SQL as the query interface. This work includes SMCQL [39], Conclave [40] and Senate [41]. SMCQL is a framework for executing SQL series over a Private Data Network (PDN) [39], where a user submits a query to an honest broker which the orchestrates the Secure Multi-Party Computation over the Private Data Network with an honest-but-curious threat model. SMCQL supports joins, aggregations and group-by queries. Conclave [40] supports a similar set of operations to SMCQL but allows weakening of its security model to achieve improved performance. Senate was developed after SMCQL and Conclave, and through the planning protocol developed – which enables more parallelisation of computation, and compartmentalisation to identify when subsets of nodes work towards a particular result – has a performance that is orders of magnitude faster than SMCQL and Conclave. Senate additionally supports a stronger malicious security guarantee.

There is work supporting SMPC over fragments of SPARQL. The Secure Framework for Graph Outsourcing and SPARQL Evaluation (GOOSE) [42] uses secure multi-party computation to achieve the following features: no cloud node can learn the graph; no cloud node can learn at the same time the query and the query answers; and, an external network observer cannot learn the graph, the query, or the query answers. However, GOOSE is limited to support Unions of Conjunctions of Regular Path Queries (UCRPQ) and does not support common numeric or build-in operations such as COUNT, SUM and AVG. Further, GOOSE requires an honest broker to design the query plan and communicate it to the compute cluster of graph databases. GOOSE also has a fixed assumption that the graph databases executing the query are honest-but-curious. That is, the databases can be trusted to execute the plan given to them by the broker.

SMPG: Secure Multi Party Computation on Graph Databases [43] has been produced as a position paper and prototype for automatically executing MPC evaluation of Cypher queries [23] over Neo4J [44] databases. SMPG is built using Conclave and so has matching weak security assumptions in addition to performance and expressivity challenges. Cypher [23] is problematic for distributed queries as identifiers are local to the database. Consequently, queries often must explicitly disambiguate entities by identifying the which node it occurs in within queries `MATCH(node1:label1)`. This is not amenable to one of our driving goals which is to abstract away all underlying architectures to the greatest extend possible and have a pure data-layer for systems to work with.

4. Research question(s) and hypotheses

This research aims to develop standardised declarative query language (data sublanguage) supporting zero-knowledge verifiable provenance statements – which shall herein be referred to as a verifiable data sublanguage.

Research Question

Which logic profiles of verifiable data sublanguages afford computationally efficient query-engine implementations against given configurations of data and identity infrastructure?

Sub Questions (RQs)

1. Which logical profiles can be supported by a query engine implementing a verifiable data sublanguage for a single graph database.
2. When implementing this verifiable data sublanguage across a distributed set of graph databases:
   a) what is the minimal set of information that can be shared (disclosed) between the graph databases in computing the result, and
   b) what logical profiles of the verifiable data sublanguage from RQ1 can be efficiently supported for given configurations of graph databases.

RQ1 is expected to use Zero Knowledge Proof (ZKP) techniques, RQ2 is expected to develop federated query planning and execution engines that apply Secure Multi-Party Computation (SMPC).

5. Preliminary results

We are currently progressing on RQ1 and have not yet started RQ2. So far, we have developed an extension of SPARQL 1.1 that supports zero-knowledge proof that a SELECT query result is correct. Specifically, we prove that results are computed from a knowledge base comprising facts signed by a given set of issuers. For the architecture we develop, facts are ingested into the knowledge base from W3C Verifiable Credentials [4] signed using the ed25519 signature algorithm [45]. We have implemented this SPARQL extension in the RISCZero Zero Knowledge Virtual Machine (ZKVM), benchmarked the implementation, and outline the performance improvements that can be made to make a production ready implementation of the SPARQL extension in a near-term timescale.

The RISCZero implementation supports all spec compliant SPARQL 1.1 SELECT Queries. We have benchmarked the implementation on a range of machines and found the execution time too slow for production evironments. For example, a Macbook Air machine running macOS Sequoia 15.4.1 with an M1 chip and 16GB of memory takes approximately 7.5 minutes to execute the query `SELECT * WHERE { ?s ?p ?o }` over a single credential containing 23 triples. This was expected as the goal of the ZKVM implementation was to design the SPARQL SELECT interface and prove the feasibility of implementing the interface.

6. Evaluation

The theoretical complexity analyses will be performed in the same manner as the SPARQL complexity analyses have been performed by Perez et. al. [46] for SPARQL and Horrocks et. al. [47] for rule-based inference profiles such as SROIQ.

For experimental evaluations we have developed zkSPARQL bench dataset[1] which consists of the Lehigh University Benchmark (LUBM) [48] and Berlin SPARQL Benchmark (BSBM) [49] with the

[1] https://github.com/jeswr/zkSPARQL-bench/

datasets partitioned and signed to form sets of credentials.

7. Reflections and future work

We have so far designed the SPARQL 1.1 SELECT interface and prove the feasibility of implementing the interface against a centralised database.

Future work is to support the ASK and CONSTRUCT query methods; and to support the TSV, CSV and XML results formats for SELECT queries. Moreover, there is also future work to support SPARQL 1.2, including the development of built-ins to support surfacing proof statements as assertions on reified triples. We also need to begin work on supporting query across distributed databases using (S)MPC.

We are also working to develop more efficient implementations SPARQL 1.1 SELECT interface - with an explicit goal of brinigng the proof time below 1 second when executing common queries over datasets with less than 1000 triples. To do this we are developing implementations that do not rely on a ZKVM. For simple queries such as BGP pattern matching we are working with Braun et al. [35] to directly use properties of BBS+ [17] signatures to generate the proof. For more complex queries, for instance requiring string operations, we are developing circuit-based implementations - which Gu et. al [28] show to be efficient in their SQL work.

Declaration on Generative AI

During the preparation of this work, the author(s) used ChatGPT, Claude in order to: discover related work, Grammar and spelling check, Paraphrase and reword. After using this tool/service, the author reviewed and edited the content as needed and takes full responsibility for the publication's content.

Acknowledgements

This student is supervised by Sir Nigel Shadbolt and Professor Jun Zhao and funded by the Department of Computer Science, University of Oxford.

References

[1] S. Harris, A. Seaborne, E. Prudĥommeaux, SPARQL 1.1 Query Language, W3C Recommendation, W3C, 2013. https://www.w3.org/TR/sparql11-query/.

[2] O. Hartig, A. Seaborne, R. Taelman, G. Williams, T. P. Tanon, Sparql 1.2 query language, https://www.w3.org/TR/sparql12-query/, 2025. W3C Working Draft.

[3] A. V. Sambra, E. Mansour, S. Hawke, M. Zereba, N. Greco, A. Ghanem, D. Zagidulin, A. Aboulnaga, T. Berners-Lee, Solid: a platform for decentralized social applications based on linked data, MIT CSAIL & Qatar Computing Research Institute, Tech. Rep. (2016).

[4] M. Sporny, D. Longley, D. Chadwick, O. Steele, Verifiable credentials data model v2.0, 2024. URL: https://www.w3.org/TR/2024/CRD-vc-data-model-2.0-20240416/.

[5] J. Wright, Disambiguating data wallets, 2025. URL: https://blog.jeswr.org/2025/02/14/data-wallets, accessed on May 31, 2025.

[6] Personal identification — ISO-compliant driving licence - Part 5: Mobile driving licence (mDL) application, 2021.

[7] European Union, Regulation (eu) no 910/2014 of the european parliament and of the council of 23 july 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing directive 1999/93/ec, 2014. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32014R0910, official Journal of the European Union, L 257, 28.8.2014, p. 73–114.

[8] European Commission, European digital identity, https://commission.europa.eu/strategy-and-policy/priorities-2019-2024/europe-fit-digital-age/european-digital-identity_en, 2025. Accessed: 2025-05-13.

[9] Data (use and access) bill [hl], https://bills.parliament.uk/bills/3825, 2024. Bill [HL] 40, introduced in the House of Lords, UK Parliament, 2024–25 session.

[10] UN/CEFACT, Un transparency protocol (untp) specification, 2025. URL: https://uncefact.github.io/spec-untp/, accessed: 2025-05-13.

[11] Cards and security devices for personal identification — Building blocks for identity management via mobile devices - Part 1: Generic system architectures of mobile eID systems, 2023.

[12] D. Fett, K. Yasuda, B. Campbell, Selective Disclosure for JWTs (SD-JWT), Internet-Draft draft-ietf-oauth-selective-disclosure-jwt-19, Internet Engineering Task Force, 2025. URL: https://datatracker.ietf.org/doc/draft-ietf-oauth-selective-disclosure-jwt/, work in Progress.

[13] M. B. Jones, J. Bradley, N. Sakimura, JSON Web Token (JWT), RFC 7519, 2015. URL: https://www.rfc-editor.org/info/rfc7519. doi:10.17487/RFC7519.

[14] M. Sporny, G. Kellogg, M. Lanthaler, D. Longley, Json-ld 1.1 - a json-based serialization for linked data, https://www.w3.org/TR/json-ld11/, 2020. W3C Recommendation.

[15] D. Wood, M. Lanthaler, R. Cyganiak, RDF 1.1 Concepts and Abstract Syntax, W3C Recommendation 25 February 2014 (2014). URL: https://www.w3.org/TR/rdf11-concepts/.

[16] Department for Science, Innovation and Technology, Government Digital Service, Department for Transport, Ministry of Defence, Office for Veterans' Affairs, Digital driving licence coming this year, 2025. URL: https://www.gov.uk/government/news/digital-driving-licence-coming-this-year, accessed: 2025-05-13.

[17] T. Looker, V. Kalos, A. Whitehead, M. Lodder, The BBS Signature Scheme, Internet-Draft draft-irtf-cfrg-bbs-signatures-08, Internet Research Task Force (IRTF), 2025. URL: https://identity.foundation/bbs-signature/draft-irtf-cfrg-bbs-signatures.html, work in Progress.

[18] D. Boneh, X. Boyen, H. Shacham, Short group signatures, in: Annual international cryptology conference, Springer, 2004, pp. 41–55.

[19] W3C Credentials Community Group, Credentials community group, https://www.w3.org/community/credentials/, 2025. Accessed: 2025-05-31.

[20] N. Drummond, R. Shearer, The open world assumption, in: eSI workshop: the closed world of databases meets the open world of the semantic web, volume 15, 2006, p. 1.

[21] J. Pérez, M. Arenas, C. Gutierrez, Semantics and complexity of sparql, in: I. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, L. M. Aroyo (Eds.), The Semantic Web -

ISWC 2006, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 30–43.
[22] C. J. Date, A Guide to the SQL Standard, Addison-Wesley Longman Publishing Co., Inc., 1989.
[23] N. Francis, A. Green, P. Guagliardo, L. Libkin, T. Lindaaker, V. Marsault, S. Plantikow, M. Rydberg, P. Selmer, A. Taylor, Cypher: An evolving query language for property graphs, in: Proceedings of the 2018 international conference on management of data, 2018, pp. 1433–1445.
[24] R. Taelman, M. Vander Sande, R. Verborgh, Graphql-ld: linked data querying with graphql, in: ISWC2018, the 17th International Semantic Web Conference, 2018, pp. 1–4.
[25] E. Ben-Sasson, A. Chiesa, E. Tromer, M. Virza, Succinct {Non-Interactive} zero knowledge for a von neumann architecture, in: 23rd USENIX Security Symposium (USENIX Security 14), 2014, pp. 781–796.
[26] M. Bellés-Muñoz, M. Isabel, J. L. Muñoz-Tapia, A. Rubio, J. Baylina, Circom: A circuit description language for building zero-knowledge applications, IEEE Transactions on Dependable and Secure Computing 20 (2022) 4733–4751.
[27] Halo2 Contributors, The halo2 book, 2025. URL: https://zcash.github.io/halo2/, accessed: 2025-05-13.
[28] B. Gu, J. Fang, F. Nawab, Poneglyphdb: Efficient non-interactive zero-knowledge proofs for arbitrary sql-query verification, Proc. ACM Manag. Data 3 (2025). URL: https://doi.org/10.1145/3709713. doi:10.1145/3709713.
[29] RISC Zero, Universal zero knowledge, 2025. URL: https://risczero.com/, accessed: 2025-05-13.
[30] D. Kanter, Risc-v offers simple, modular isa, Microprocessor Report 1 (2016) 1–5.
[31] T. Liu, Z. Zhang, Y. Zhang, W. Hu, Y. Zhang, Ceno: Non-uniform, segment and parallel zero-knowledge virtual machine, Journal of Cryptology 38 (2025) 17.
[32] U. Roy, Introducing sp1: A performant, 100% open-source, contributor-friendly zkvm, 2024. URL: https://blog.succinct.xyz/introducing-sp1/, accessed: 2025-05-13.
[33] Powdr Contributors, Powdr documentation, 2025. URL: https://docs.powdr.org, accessed: 2025-05-13.
[34] ZKM Contributors, Zkm architecture, 2025. URL: https://docs.zkm.io/zkm-architecture, accessed: 2025-05-13.
[35] C. H.-J. Braun, T. Käfer, RDF-based Semantics for Selective Disclosure and Zero-knowledge Proofs on Verifiable Credentials, in: Proceedings of the 21st Extended Semantic Web Conference (ESWC 2025), CEUR Workshop Proceedings, CEUR-WS.org, 2025. URL: https://2025.eswc-conferences.org/, to appear.
[36] A. C.-C. Yao, How to generate and exchange secrets, in: 27th annual symposium on foundations of computer science (Sfcs 1986), IEEE, 1986, pp. 162–167.
[37] M. Silaghi, Smc tutorial, https://cs.fit.edu/~msilaghi/SMC/tutorial.htm, ???? Accessed: 2025-05-18.
[38] A. Rastogi, N. Swamy, M. Hicks, Wys*: a dsl for verified secure multi-party computations, in: International Conference on Principles of Security and Trust, Springer, 2019, pp. 99–122.
[39] J. Bater, G. Elliott, C. Eggen, S. Goel, A. Kho, J. Rogers, Smcql: secure querying for federated databases, Proc. VLDB Endow. 10 (2017) 673–684. URL: https://doi.org/10.14778/3055330.3055334. doi:10.14778/3055330.3055334.
[40] N. Volgushev, M. Schwarzkopf, B. Getchell, M. Varia, A. Lapets, A. Bestavros, Conclave: secure multi-party computation on big data, in: Proceedings of the Fourteenth EuroSys Conference 2019, 2019, pp. 1–18.
[41] R. Poddar, S. Kalra, A. Yanai, R. Deng, R. A. Popa, J. M. Hellerstein, Senate: a {Maliciously-Secure}{MPC} platform for collaborative analytics, in: 30th USENIX Security Symposium (USENIX Security 21), 2021, pp. 2129–2146.
[42] R. Ciucanu, P. Lafourcade, : A secure framework for graph outsourcing and sparql evaluation, in: IFIP Annual Conference on Data and Applications Security and Privacy, Springer, 2020, pp. 347–366.
[43] N. Al-Juaid, A. Lisitsa, S. Schewe, Smpg: Secure multi party computation on graph databases., in: ICISSP, 2022, pp. 463–471.
[44] J. J. Miller, Graph database applications and concepts with neo4j, in: Proceedings of the southern

association for information systems conference, Atlanta, GA, USA, volume 2324, 2013, pp. 141–147.
[45] M. Sporny, D. Longley, G. Bernstein, Data integrity ecdsa cryptosuites v1.0, 2025. URL: https://www.w3.org/TR/vc-di-ecdsa/.
[46] J. Pérez, M. Arenas, C. Gutierrez, Semantics and complexity of sparql, in: International semantic web conference, Springer, 2006, pp. 30–43.
[47] I. Horrocks, O. Kutz, U. Sattler, The even more irresistible sroiq., Kr 6 (2006) 57–67.
[48] Y. Guo, Z. Pan, J. Heflin, Lubm: A benchmark for owl knowledge base systems, Journal of Web Semantics 3 (2005) 158–182.
[49] C. Bizer, A. Schultz, The berlin sparql benchmark, International Journal on Semantic Web and Information Systems (IJSWIS) 5 (2009) 1–24.