A SPARQL extension for Zero Knowledge Query over Verifiable Credentials

No Author Given

No Institute Given

Abstract. This work develops 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 signed using the ed25519 signature algorithm. We implement this SPARQL extension using a Zero Knowledge Virtual Machine, benchmark 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.
This work has immediate applications, particularly for improving the privacy of EU and UK citizens using Digital Verifiable Credentials – which are being rolled out under the European Digital Identity (EUDI) and Digital Verification Schemes (DVS) respectively.

Keywords: Verifiable Credentials · SPARQL · Zero Knowledge · Zero Knowledge Proof · RDF · JSON-LD .

1 Introduction and Background

1.1 Verifiable Credentials

Overview Verifiable Credentials (VCs) [38] are seeing widespread adoption. Driven by governments and regulation - many Australian states have had digital drivers licenses [1] since 2016; in Europe eIDAS (Electronic Identification, Authentication, and Trust Services) 2 regulation [15] is mandating that all EU citizens will have access to at least one European Digital Identity (EUDI) wallet [14]; and the UK's Data (Use and Access) Bill [3] 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) [39] - for instance - uses Verifiable Credentials to prevent conterfeiting, support sustainiability audits and improve automated decision making.

Specifications 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

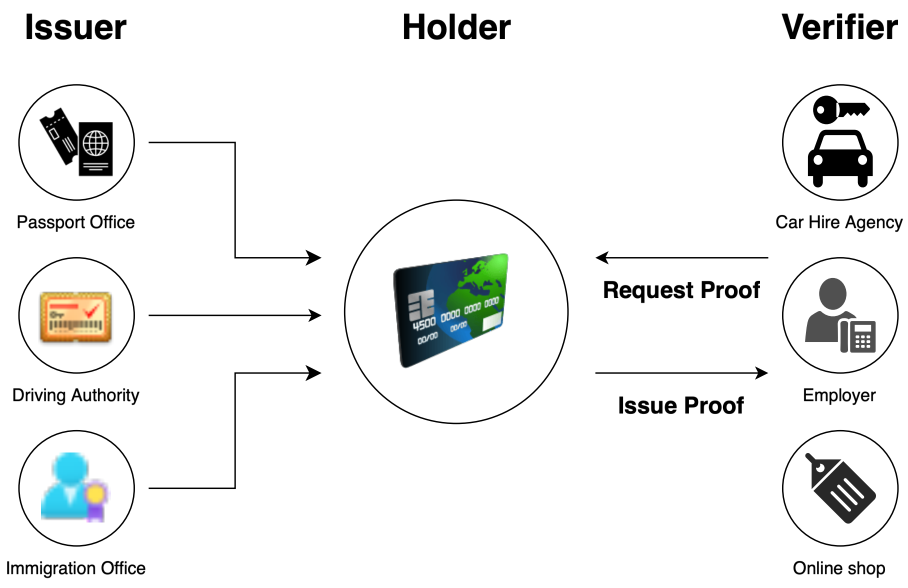

Fig. 1. The typical relationship between issuers, holders and verifiers in Verifiable Credential flows

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. 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 [1] and the Cards and security devices for personal identification (ISO23220) specification [2]. 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 [16] - based on JSON Web Tokens (JWT's) [23] 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 [38] provides a general model for adding integrity proofs to JSON-LD [36] documents. This standard does not restrict what statements can be made within the credential - and by requiring the use of JSON-LD - supports RDF [40] and formal semantics out-of-the-box.

Selective Disclosure in Verifiable Credentials 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 [12]. SD is typically implemented using the BBS signature scheme [28]. 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 [9].

1.2 RDF, SPARQL and the Semantic Web

The Semantic Web [6] is a technology stack supported by the World Wide Web Consortium (W3C) to enable interoperability between systems. At its core is the concept of Linked Data, which is the "collection of interrelated [and machine readable] datasets on the Web." Linked Data is achieved by publishing and exchanging data in the standardised Resource Description Framework (RDF) [40].

The atom of a piece of data in RDF is called a triple. A triple relates a resource (a subject) to another resource or value (an object) through a predicate (a verb). A knowledge graph is a set of triples that link and describe such resources and entities, and the triple can be thought of as an edge in the graph. Vocabularies (or ontologies) are used to define concepts and relationships (predicates) in RDF datasets.

The SPARQL Protocol and RDF Query Language (SPARQL) [20] is a standardised query language to query over RDF data.

1.3 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) [5]. zkSNARKs allow provers (holders) to generate a full proof without interacting with the verifier. SD proofs generated from BBS Signatures [9] are zkSNARKs.

Zero Knowledge Arithmetic Circuits builders (ZK Circuits) [4] - such as circom [4], plonky3 and Halo 2 [19] 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 [4] supporting quadratic constraints. Gu et. al [17] prove that circuit builders can be used to generate zkSNARKs of correct execution of SQL queries with their implementation of PoneglyphDB [17] using the Halo 2 Circuit Compiler [19].

Zero Knowledge Virtual Machines (ZKVMs) enable proof of correct execution of a given instruction set. RISC Zero [32] 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 [24] - 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 [26], SP1 [33], Nexus21, Powdr [31] and ZkMIPS [41].

zkSNARKs for RDF Datasets - Braun et al. [10] 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. [10] - 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.

1.4 zkSNARKs over W3C VCs

Verifiale Credentials are widely adopted, and support for Selective Disclosure indicates a demand for credential *holders* to support data minimization. Selective Disclosure 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` [1] rather than holders deriving this statement from a date of birth.

Using the capabilities of contemporary zkSNARKs - this paper advances the data minimisation capabilities of credential *holders*: enabling *verifiers* to issue SPARQL 1.1 [20] SELECT queries and *holders* to reveal only the result as evaluated over one or more W3C Verifiable Credentials and the public keys of the credential issuers. This paper addresses only W3C Verifiable Credentials, as it is the only standard to support RDF natively.

2 Query interface design

We present an interface for adding zkSNARK proofs to SPARQL SELECT results to prove that results are derived from statements made by a given set of issuers. An example of this response we describe below can be found at anonymous.4open.science[1].

[1] https://anonymous.4open.science/r/risc0-ed25519-zk-sparql-4FFB/sparql_result_minimal.json

This interface requires that SELECT queries use the `application/sparql-results+json` query results format; and adds to the JSON object response a `proof` entry which is to be used to add a proof that the SPARQL results are true. There are two entries that MUST be present in this proof object. The first is a `type` key which is used to signal to a verifier of the type of proof that has been applied to the SPARQL results - and in turn how the proof can be verified.

The second required field is a `keys` entry which is a list of objects consisting of the `id` and `value` of the set of publicKeys used to validate the signatures of credentials passed to the ZKVM. The `id` can be used to dereference the Controller Identifier Document (CID) [35] in which the `publicKey` is defined. This enables the verifier to check both that the publicKey does correspond to the claimed `id` - and discover who is the data *issuer* that has created the `publicKey`.

Regardless of the `type` of the proof, and associated additional values in the proof object - the proof MUST justify that each result is a sound entailment of data signed by the set of `publicKey`s included in the output proof object. The proof need not provide any completeness guarantees. This aligns with the Open World Assumption (OWA) [25] in that the query engine may be operating over incomplete facts or a knowledge base with missing credentials regardless. Note that this means that in the case of an empty results set, no proof is required. In this case the type can be set to `none`, the `keys` entry may have an empty array, and no further entries are required. Proof methods (`type`s) MAY not provide proof of required relationships between separate results sets. For instance, the proof MAY not guarantee that an `ORDER BY` has been correctly accounted for in the query result. It should be known from the proof `type` whether or not thisholds.

3 Implementing the Query interface using the RISC Zero ZKVM

.

This paper presents one of the simplest possible implementations of the query interface outlined in Section 2. This proves the feasibility of implementing the interface and provides a basline for benchmarks against which future implementations can be evaluated. An architectural diagram of this implementation is presented in Figure 2. An architectural diagram of a verifier for this interface is presented in Figure 3.

RISC Zero was selected as the ZKVM with which to implement this architecture as the authors found it to have the easiest abstractions to use at the time of writing. RISC Zero [13] is not the most performant ZKVM currently available.

3.1 Architecture

Prover Architecture A set of credentials compliant with the W3C Verifiable Credentials 2.0 Data Model [38] and signed using the ecdsa-rdfc-2019 cryptosuite as specified by the Data Integrity ECDSA Cryptosuites v1.0 [37] are accepted as

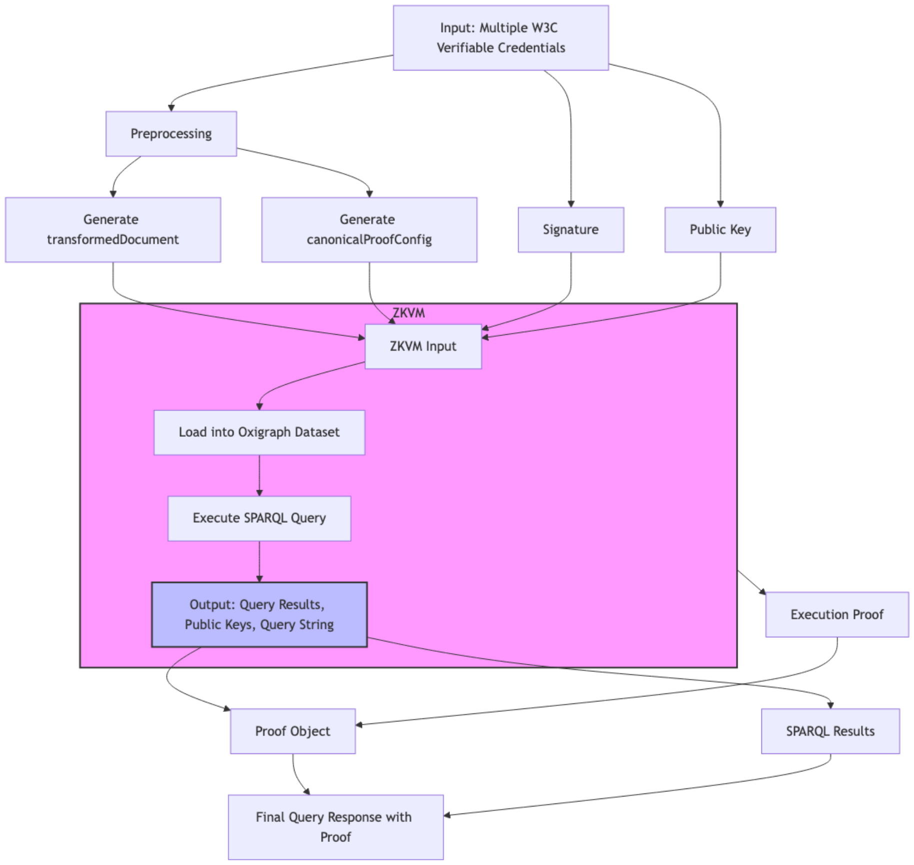
Input: Multiple W3C Verifiable Credentials
Preprocessing
Generate transformedDocument
Generate canonicalProofConfig
Signature
Public Key
ZKVM
ZKVM Input
Load into Oxigraph Dataset
Execute SPARQL Query
Output: Query Results, Public Keys, Query String
Execution Proof
Proof Object
SPARQL Results
Final Query Response with Proof

Fig. 2. Architecture of the Prover implemented using the RISC Zero ZKVM

input. These credentials are preprocessed outside of the Zero Knowledge Virtual Machine (ZKVM) to - for each credential - generate a `transformedDocument` and `canonicalProofConfig`. These are canonicalised N-Quad [11] representations of the credential document, and proof configuration respectively. Canonicalisation is performed using the RDF Dataset Canonicalization [27] algorithm - as required for proof verification in Section *3.2.2 Verify Proof (ecdsa-rdfc-2019)* of the Data Integrity ECDSA Cryptosuites v1.0 [37]. This is performed using `ed25519-verify-preprocess` command from `vc-cli.js`[2].

The ZKVM then takes as input - for each credential - the `transformedDocument`, `canonicalProofConfig`, `verificationMethod.proofValue` of the credential and `publicKey` of the credential issuer. The `publicKey` is discoverable from the Controller Identifier Document (CID) [35] of the credential issuer. The CID can be dereferenced by resolving the `verificationMethod.id` Decentralised Identifier

[2] https://anonymous.4open.science/r/vc-cli-js-BCC8/

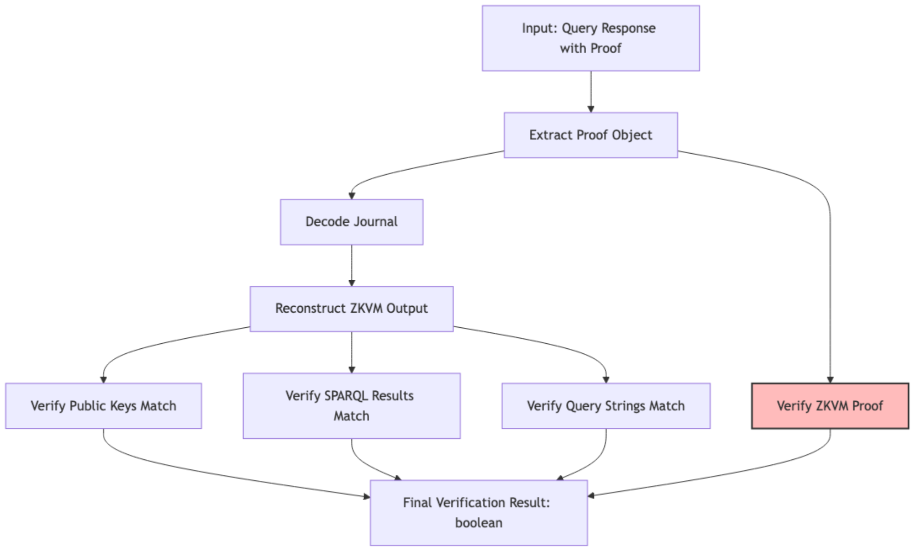
Input: Query Response with Proof
Extract Proof Object
Decode Journal
Reconstruct ZKVM Output
Verify Public Keys Match
Verify SPARQL Results Match
Verify Query Strings Match
Verify ZKVM Proof
Final Verification Result: boolean

Fig. 3. Architecture of the Verifier implemented using the RISC Zero ZKVM

(DID) [34] value. Additionally, a SPARQL [20] SELECT query is supplied as input to the ZKVM.

Within the ZKVM the ecdsa-rdfc-2019 Proof Verification algorithm [37] is first applied to each set of these inputs - the ZKVM will terminate with an error if an invalid proof is present. The `transformedDocument` of each credential is then loaded into an Oxigraph [30] `Dataset` - with blank node renaming performed to ensure that blank nodes from distinct documents are named differently. The SPARQL SELECT query provided to the ZKVM is then executed against the dataset.

The ZKVM returns the query response as a string containing the results in a `application/sparql-results+json` format, the original query as a string, and a list of the `publicKeys` used to verify the signature of each preprocessed credential.

Outside of the ZKVM, the query response is parsed into JSON. A `proof` object is added to this JSON object to produce the query response with proof described in Section 2. Particularly, this proof has `type` of "Risc0ZKVM" and has the following custom fields:

1. `inner` the proof object showing correct execution of the ZKVM
2. `zkvm_id` an array of integers identifying the guest program that was used by the ZKVM
3. `keys` an array of objects with the `id` and `value` of `publicKeys` used to validate the signatures of credentials passed to the ZKVM.
4. `journal` a hex-encoded string of the journal[3] of the RISC Zero ZKVM

[3] https://dev.risczero.com/terminology#journal

5. `query` a string of the query that was executed in the ZKVM

The ZKVM functionality is available at `risc0-ed25519-zk-sparql-4FFB` on anonymous.4open.science[4].

Verifier Architecture The role of the verifier is to check the validity of a given set of SPARQL results with a proof object.

The verification process first extracts and removes the proof object from the results. It then decodes the journal from hex format and reconstructs the output that was produced by the ZKVM. This reconstructed output is compared against the provided results to ensure they match exactly. Specifically, it verifies that:

1. The public keys used for verification match those in the proof
2. The SPARQL results match those produced by the RISC Zero ZKVM
3. The query string matches the one executed in the RISC Zero ZKVM

Finally, the verifier uses the RISC Zero ZKVM's verification functionality to verify the proof of correct execution. This ensures that the results were indeed computed by the correct program running in the ZKVM, and that the computation was performed correctly.

3.2 Choice of cryptosuites

W3C Verifiable Credentials support a range of Securing Mechanisms - grouped into two classes: enveloping proofs, and embedded proofs. Enveloping Proofs, such as those defined by *Securing Verifiable Credentials using JOSE and COSE* [22] wrap and sign the entire JSON object.

The only two specifications that currently support signature of the RDF Dataset - rather than JSON-LD presentation - are the *Data Integrity BBS Cryptosuites v1.0* [7] and the Data Integrity ECDSA Cryptosuites v1.0 [37]. Both are embedded proofs. When signing the dataset, the suites will either sign the RDFC14N Canonicalised Hash [27] of the dataset; or hashes of individual quads that have been preprocessed using the RDFC14N Canonicalisation algorithm.

Since we are treating credentials as an RDF Dataset, we require a proof mechanism that signs the RDF Dataset rather than the JSON representation of the credential. Consequently, we must choose between these two cryptographic suites for our experiments. A further justification for selecting a cryptographic suite that signs an N-Quads [11] representation of the data is that the N-Quads represenation can be input and parsed within the ZKVM. This eliminates the need to perform expensive JSON-LD [36] processing within the ZKVM.

The selective disclosure capabilities of the BBS Cryptosuite do not add value for the RISC Zero ZKVM implementation of the query interface - as the architecture simply uses the signature to verify the integrity of all facts within the ZKVM.

[4] https://anonymous.4open.science/r/risc0-ed25519-zk-sparql-4FFB/

RISC Zero has a set of pre-compiled libraries for generating and validating signatures. It has such a library specifically for ECDSA [21] signature, but not currently for BBS [9] signatures. Consequently, we choose to use the ECDSA signature in order to minimise the signature verification time within the ZKVM.

3.3 Validation Gaps

Whilst we perform integrity checking within the ZKVM, in this paper we do not perform other validity checks expected by the W3C Verifiable Credentials 2.0 Data Model [38]. Specifically we do not currently use the `validFrom` and `validUntil` properties to check if a credential is currently valid - or derive `validFrom` and `validUntil` bounds for the result of the SPARQL query. We also do not check the `credentialStatus` which means that data from a credential can be used even if the credential has been revoked by an issuer. We present resolutions in Section 6.

3.4 Security Considerations

In addition to the high performance cost of the RISC Zero ZKVM implementation of the SPARQL query interface specified in Section 2 - there are several security issues that arise with the current implementation.

The first security consideration that should be accounted for is that **bugs in the rust code and dependencies may provide attack vectors to produce false facts**. In particular any known or discovered bugs in Oxigraph [30] could be exploited to enable the production of query results that are not validly entailed from a set of input credentials. Further, if there are any bugs present in the cryptography library used to verify the signature on the credential; this may be exploited to enable corrupted or malicious credentials to pass the validity check and be used in the evaluation of the SPARQL query results.

Similarly, we are **entirely dependent on RISC Zeros' infrastructure** for generating the proof of correct execution and validating that proof. There is currently no standard defining the expected content of proofs to enable independent implementations of ZKVMs which can each independently generate and validate proofs. This means that if there are any bugs in RISC Zero then there is a risk that invalid proofs can be verified as true. The RISC Zero prover does claim to be quantum safe.

Further, the `Ed25519` signature scheme used for signing the digital credentials in the experiment is not considered quantum safe.

4 Experiments

4.1 Benchmark

We have developed a bespoke dataset for evaluating the performance this implementation of the SPARQL extension for Zero Knowledge Query over verifiable credentials.

```
SELECT DISTINCT ?s ?givenName ?familyName
WHERE {
    ?s citizenship:employmentAuthorizationDocument [
        citizenship:lprCategory "C09" ] ;
    citizenship:residentSince ?date ;
    schema:birthDate ?birthDate ;
    schema:givenName ?givenName ;
    schema:familyName ?familyName ;
    vdl:license [ vdl:expiryDate ?expiryDate ] .

  FILTER( ?date < xsd:dateTime("2020-01-01T00:00:00") )
  FILTER( ?birthDate < xsd:dateTime("2007-01-01T00:00:00") )
  FILTER( ?expiryDate > xsd:dateTime("2025-01-01T00:00:00") )
}
```

Fig. 4. Can Drive query

```
SELECT DISTINCT ?person ?givenName ?familyName
WHERE {
  ?person a citizenship:EmployablePerson ;
    schema:givenName ?givenName ;
    schema:familyName ?familyName .
}
ORDER BY ?name
```

Fig. 5. Employment query

There are two requirements for this dataset. First it must consist of credentials validly signed using the ecdsa-rdfc-2019 Proof Verification algorithm [37], secondly the dataset must be small ($\leq$ 1000 triples, $\leq$ 10 credentials) due to high time complexity of proof generation with the RISC Zero ZKVM. The second requirement prevents benckmarking against the zkSPARQL bench dataset[5] which consists of the Lehigh University Benchmark (LUBM) [18] and Berlin SPARQL Benchmark (BSBM) [8] with the datasets partitioned and signed to form sets of credentials.

Datasets There are four credentials we use for our datasets[6] - all signed by distinct issuers: a drivers license (23 triples without proof); a bachelors degree credential (15 triples without proof); an employment authorisation credential (20 triples without proof); and a residence card credential (21 triples without proof). The first dataset (**4-credentials**) contains all credentials, has 79 triples and four integrity proofs. The second dataset (**1-credential**) consists of the drivers license credential *only* and has 23 triples and one integrity proof.

[5] https://anonymous.4open.science/r/zkSPARQL-bench-56DE/
[6] https://anonymous.4open.science/r/vc-generated-example-A21E/generated/ed25519/

Queries Three distinct SPARQL [20] SELECT queries are used for benchmarking[7]. They are **SELECT ALL** - which selects the subject, predicate and object of all triples in the dataset; **Can Drive** (c.f. Fig 4) - which selects the ID, first, and last name of all adults with a valid residency and drivers license - using seven joins and three filters; and **Employment** (c.f. Fig 5) - which selects the ID, first, and last name of all `EmployablePersons` - using two joins and one `OrderBy` statement.

4.2 Results

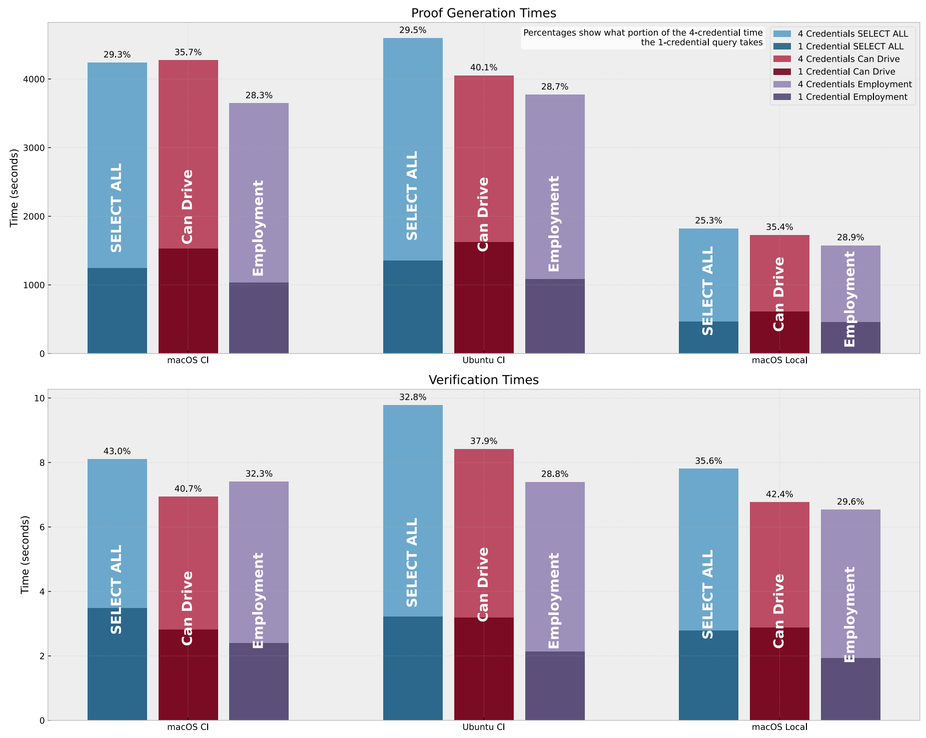

Fig. 6. Benchmark results showing the performance of the SPARQL extension in terms of query time and proof size.

We perform benchmarking on the three following machines: **macOS CI** - the default MacOS GitHub runner which runs MacOS 15, has 3 M1 cores and 16GB of memory; **Linux CI** - the default Linux (Ubuntu) GitHub runner which has 4 cores and 16GB of memory; and **macOS Local** - a Macbook Air machine

[7] https://anonymous.4open.science/r/risc0-ed25519-zk-sparql-4FFB/queries/

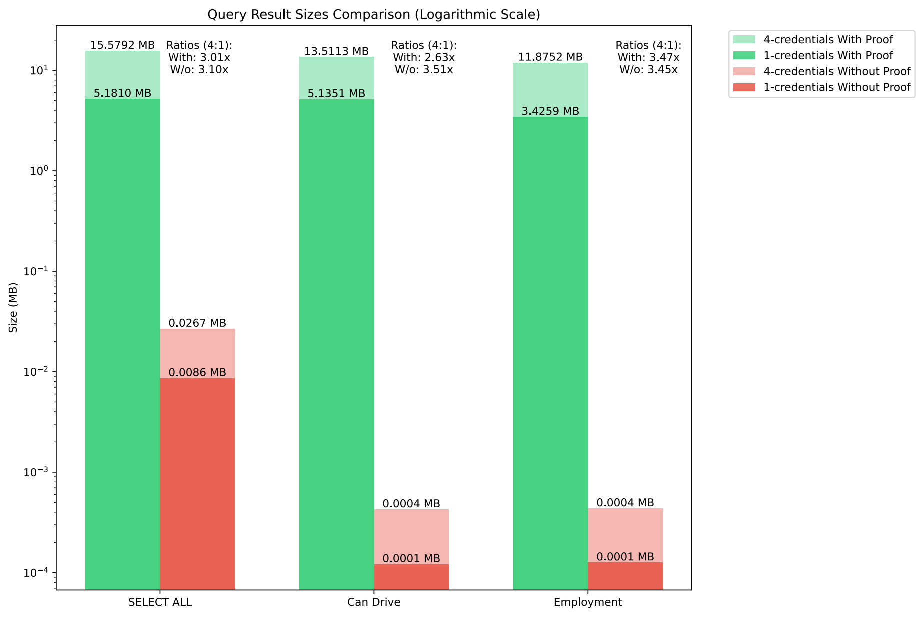

Fig. 7. Sizes of the results object for each query performed with the **1-credential** and **4-credentials** datasets

running macOS Sequoia 15.4.1 with an M1 chip and 16GB of memory. All experiments use Rust v1.86.0, LLVM v19.1.7 and RISC Zero v2.0.2. All versioning of dependent rust packages in this project is specified in the projects' lockfile[8]. The results of the experiment are presented in Figure 6. Figure 7 shows the size of the results object for each query performed with the **1-credential** and **4-credentials** dataset.

Additionally we tested the minimal SELECT ALL query on the MacOS local environment without proof verification - finding the ZKVM ran for 255s - as compared to the 459s taken to perform proof verification. This means the verification of a single signature takes approximately 204s on a local MacOS machine.

4.3 Analysis

The first aim of performing this benchmark is to provide a baseline set of performance metrics against which to evaluate future implementations of the interface defined in Section 2. The second aim is to evaluate whether this implementation is performant enough to be deployed at production scale. The third aim is to identify areas to improve the performance of the architecture defined in Section 3.

[8] https://anonymous.4open.science/r/risc0-ed25519-zk-sparql-4FFB/Cargo.lock

The first aim requires no further discussion. To the second aim - since *holder* applications are often consumer facing applications such as wallets - we set the threshold of production-readyness as being when most SELECT queries can be executed over several credentials on consumer hardware in less than one second. One second is chosen as the threshold because this is a time frame in which a user's flow of thought remains uninterrupted [29]. The implementation is clearly not production ready by this measure, given that a select all query over a single credentail takes over 8 minutes with this implementation.

As a performance comparison, we evaluate these dataset and queries on Oxigraph outside of the RISC0 ZKVM environment. On the MacOS local machine we found that the time to load and execute the query was between 0.15 - 0.25s for all datasets and queries. Specifically for the 1-credential dataset, the SELECT ALL, Can Drive and Employment queries took 0.18s, 0.17s and 0.17s respectively; and on the 4-credential dataset, the queries took 0.18s, 0.23s and 0.17s. Consequently, the time for veryifying signatures and evaluating queries within the ZKVM is $\approx$ 2.5k times slower than evaluating queries outside of the ZKVM.

All remaining analysis is thus performed with a goal of understanding where the most compute was consumed during experiments - so as to improve the performance of the RISC Zero implementation. First we note that for a given dataset, the same integrity check(s) are being performed across each query. This means that only the parsing, evaluation and serialisation of a query will affect the query performance on a given environment. We observe the following trends in the data that was collected. For the **SELECT ALL** and **Employment** queries - which have fewer join and filter operation than the `Can Drive` query - the evaluation time of the **1-credential** dataset is 25-30% that of the evaluation time over the **4-credential** dataset. This indicates that proof time scales at approximately a linear rate for these queries. For the `Can Drive` query - which is the most complex query - the **1-credential** dataset takes 35-40% of the evaluation time over the **4-credential** dataset - indicating that the time complexity with respect to dataset size is sublinear. The flamegraph breakdown[9] helps explain this phenomenon - jointly the query parsing and optimisation steps; which are constant time operations take notably longer than query evaluation. This points to a potential optimisation being parsing the query outside of the ZKVM and passing in the already tokenised and optimised query.

5 Conclusion

This paper presents an interface for adding zkSNARK proofs to SPARQL SELECT results to prove that results are derived from statements made by a given set of issuers, and proved that it is possible to implement this interface using the RISC Zero ZKVM. The implementation was benchmarked against datasets ranging between one to four credentials (23 to 70 triples) in size.

The interface we develop has immediate applications, particularly for improving the privacy of EU and UK citizens using Digital Verifiable Credentials –

[9] https://anonymous.4open.science/r/risc0-ed25519-zk-sparql-4FFB/flamegraphs/

which are being rolled out under the European Digital Identity (EUDI) and Digital Verification Schemes (DVS) respectively. However, future work - as outlined in Section 6 - is required to improve the performance and security guarantees of available implementations.

6 Future Work

The primary aim of this paper was to design the SPARQL SELECT interface and prove the feasibility of implementing the interface.

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. Additionally, there is future work to support all signature schemes used by credential issuers so as to ensure *holders* can execute queries over any credentials they receive. As outlined in Section 3.3 the interface and implementation developed do not currently support some validity checks; future work will ensure validity checks are performed in generating the result or appropriate information is added to the proof object to enable the verifier to perform this check.

The following work can be done to develop more performant implementations of the existing query interface. Order is by expected ease of implementation:

1. Filtering out irrelevant credentials: In the experiments performed in Section 4, the time taken for SPARQL proof generation was approximately proportional to the number of credentials given as input to the ZKVM. Currently, all credentials are passed to the ZKVM even if no triples from a credential are used in evaluating a query result. The time taken to verify the credential signatures, and evaluate the query is several orders of magnitude longer than the time required to evaluate a query over the same dataset outside of the ZKVM. A pre-processing step may be added to first identify which triples; and consequently which credentials, are required - and present only those required credentials to the ZKVM. Further, this would elimate the need to invoke the ZKVM when the results set is empty.
2. Proof composition: RISC Zero has support for proof composition[10] which enables the signature verification check only be performed once per credential; and for proof of this check to be re-used across multiple queries.
3. Proof checking inside ZKVM: Rather than performing full query evaluation inside the ZKVM, we can instead evaluate the query outside the ZKVM with a proof / provenance trail of how the query was evaluated that can be verified inside the ZKVM.
4. Don't prove properties that are clear from the result inside the ZKVM: For instance, ORDER BY over variables in the result should not be proven within the ZKVM as these can be checked directly by the verifier.

[10] https://dev.risczero.com/api/zkvm/composition

5. RDF Native Operations: As discussed in Section 1.3, an extension of the work by Braun et al. [10] enables direct proof that a BGP pattern is satisfiable by the contents of a set of credentials. This means that for any SPARQL query consistent only of BGP joins it would be possible to directly prove that the BGP pattern holds with the selected variables. The support for proof of numeric bounds means this should also be feasible for queries which filter based on numeric (in)equalities. We expect this to be several orders of magnitude more performant than the baseline ZKVM implementation we present given the subsecond proof times shown in [10].
6. Implementation using a circuit builder: Gu et. al [17] demonstrate that it is possible to prove correct execution of SQL queries using the Halo 2 circuit builder. In their benchmarks, they show that proving correct execution of a query with 2 joins, 3 filters, 1 aggregation, groupby operations and orderby operations takes 161 seconds when evaluated over 60k rows. Given the size of the dataset is substantially larger than those datasets used in this paper - we expect to see significant performance gains when using a circuit builder to prove correct execution of the SPARQL query.

There is also work to be done to reduce the size of the proof object to make it easier to transmit and store a large number of such results. All of the above performance improvements which eliminate the use of a ZKVM will also affect proof size. In addition we shall evaluate the effect of using succinct RISC Zero proofs on the proof size and the provers performance. We do note the caveat that succinct proofs in RISC Zero are not currently quantum safe.

More extensive and systematic benchmarking is also required. For baseline comparison queries should also be run with a traditional query engine that is not proving query correctness. The zkSPARQL bench dataset should also be extended to have modified versions of the LUBM and BSBM benchmark which have a small number of credentials such that this ZKVM implementation can be benchmarked against those datasets.

In Section 3.4 we outlined areas of potential security vulnerablity in our work - as this work is further developed a formal security analysis should be performed for each proof `type`.

Supplimentary Material Statement The full RISC Zero ZKVM implementation described in this paper can be found on anonymous.4open.science[11] along with instructions to re-run the benchmark. Additionally the source code to generate and preprocess the credentials used in the dataset can be found on anonymous.4open.science[12] as can the code to generate the queries can be found at anonymous.4open.science[13]. The source code to generate the zkSPARQL Bench benchmark can be found at anonymous.4open.science.[14]

[11] https://anonymous.4open.science/r/risc0-ed25519-zk-sparql-4FFB/
[12] https://anonymous.4open.science/r/vc-cli-js-BCC8/
[13] https://anonymous.4open.science/r/vc-queries-js-C342/
[14] https://anonymous.4open.science/r/zkSPARQL-bench-56DE/

References

1. Personal identification — ISO-compliant driving licence - Part 5: Mobile driving licence (mDL) application (2021)
2. 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)
3. 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
4. Bellés-Muñoz, M., Isabel, M., Muñoz-Tapia, J.L., Rubio, A., Baylina, J.: Circom: A circuit description language for building zero-knowledge applications. IEEE Transactions on Dependable and Secure Computing **20**(6), 4733–4751 (2022)
5. Ben-Sasson, E., Chiesa, A., Tromer, E., Virza, M.: Succinct {Non-Interactive} zero knowledge for a von neumann architecture. In: 23rd USENIX Security Symposium (USENIX Security 14). pp. 781–796 (2014)
6. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American (2001)
7. Bernstein, G., Sporny, M.: Data integrity bbs cryptosuites v1.0. W3C Proposed Recommendation, W3C (April 2025), https://www.w3.org/TR/vc-jose-cose/
8. Bizer, C., Schultz, A.: The berlin sparql benchmark. International Journal on Semantic Web and Information Systems (IJSWIS) **5**(2), 1–24 (2009)
9. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Annual international cryptology conference. pp. 41–55. Springer (2004)
10. Braun, C.H.J., Käfer, T.: 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), https://2025.eswc-conferences.org/, to appear
11. Carothers, G.: Rdf 1.1 n-quads. W3C Recommendation, W3C (Feb 2014), https://www.w3.org/TR/n-quads/
12. 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 (Jan 2025), https://www.gov.uk/government/news/digital-driving-licence-coming-this-year, accessed: 2025-05-13
13. Ernstberger, J., Chaliasos, S., Kadianakis, G., Steinhorst, S., Jovanovic, P., Gervais, A., Livshits, B., Orrù, M.: zk-bench: A toolset for comparative evaluation and performance benchmarking of snarks. In: International Conference on Security and Cryptography for Networks. pp. 46–72. Springer (2024)
14. European Commission: European digital identity. https://commission.europa.eu/strategy-and-policy/priorities-2019-2024/europe-fit-digital-age/european-digital-identity$_en(2025), accessed : 2025 - 05 - 13$
15. 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), 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
16. Fett, D., Yasuda, K., Campbell, B.: Selective disclosure for jwts (sd-jwt). Internet-Draft draft-ietf-oauth-selective-disclosure-jwt-19, Internet Engineering Task Force (May 2025), https://datatracker.ietf.org/doc/draft-ietf-oauth-selective-disclosure-jwt/, work in Progress

17. Gu, B., Fang, J., Nawab, F.: Poneglyphdb: Efficient non-interactive zero-knowledge proofs for arbitrary sql-query verification. Proc. ACM Manag. Data **3**(1) (Feb 2025). https://doi.org/10.1145/3709713, https://doi.org/10.1145/3709713
18. Guo, Y., Pan, Z., Heflin, J.: Lubm: A benchmark for owl knowledge base systems. Journal of Web Semantics **3**(2-3), 158–182 (2005)
19. Halo2 Contributors: The halo2 book (2025), https://zcash.github.io/halo2/, accessed: 2025-05-13
20. Harris, S., Seaborne, A., Prudh́ommeaux, E.: Sparql 1.1 query language. W3C Recommendation, W3C (March 2013), https://www.w3.org/TR/sparql11-query/
21. Johnson, D., Menezes, A., Vanstone, S.: The elliptic curve digital signature algorithm (ecdsa). International journal of information security **1**, 36–63 (2001)
22. Jones, M., Prorock, M., Cohen, G.: Securing verifiable credentials using jose and cose. W3C Proposed Recommendation, W3C (March 2025), https://www.w3.org/TR/vc-jose-cose/
23. Jones, M.B., Bradley, J., Sakimura, N.: JSON Web Token (JWT). RFC 7519 (May 2015). https://doi.org/10.17487/RFC7519, https://www.rfc-editor.org/info/rfc7519
24. Kanter, D.: Risc-v offers simple, modular isa. Microprocessor Report **1**, 1–5 (2016)
25. Keet, C.M.: Open World Assumption, pp. 1567–1567. Springer New York, New York, NY (2013). https://doi.org/10.1007/978-1-4419-9863-7_734, https://doi.org/10.1007/978-1-4419-9863-7$_7$34
26. Liu, T., Zhang, Z., Zhang, Y., Hu, W., Zhang, Y.: Ceno: Non-uniform, segment and parallel zero-knowledge virtual machine. Journal of Cryptology **38**(2), 17 (2025)
27. Longley, D., Kellogg, G., Yamamoto, D., Sporny, M.: Rdf dataset canonicalization. W3C Recommendation, W3C (May 2024), https://www.w3.org/TR/rdf-canon/
28. Looker, T., Kalos, V., Whitehead, A., Lodder, M.: The bbs signature scheme. Internet-Draft draft-irtf-cfrg-bbs-signatures-08, Internet Research Task Force (IRTF) (March 2025), https://identity.foundation/bbs-signature/draft-irtf-cfrg-bbs-signatures.html, work in Progress
29. Miller, R.B.: Response time in man-computer conversational transactions. In: Proceedings of the December 9-11, 1968, fall joint computer conference, part I. pp. 267–277 (1968)
30. Pellissier Tanon, T.: Oxigraph. https://doi.org/10.5281/zenodo.7408022, https://github.com/oxigraph/oxigraph
31. Powdr Contributors: Powdr documentation (2025), https://docs.powdr.org, accessed: 2025-05-13
32. RISC Zero: Universal zero knowledge (2025), https://risczero.com/, accessed: 2025-05-13
33. Roy, U.: Introducing sp1: A performant, 100% open-source, contributor-friendly zkvm (Feb 2024), https://blog.succinct.xyz/introducing-sp1/, accessed: 2025-05-13
34. Sporny, M., Guy, A., Sabadello, M., Reed, D., Longley, D., Reed, D., Steele, O., and, C.A.: Decentralized identifiers (dids) v1.0. W3C Proposed Recommendation, W3C (July 2022), https://www.w3.org/TR/did-1.0/
35. Sporny, M., Jones, M.B., Longley, D., Sabadello, M., Reed, D., Steele, O., Allen, C.: Controlled identifiers v1.0. W3C Proposed Recommendation, W3C (March 2025), https://www.w3.org/TR/cid-1.0/
36. Sporny, M., Kellogg, G., Lanthaler, M., Longley, D.: Json-ld 1.1 - a json-based serialization for linked data. https://www.w3.org/TR/json-ld11/ (July 2020), w3C Recommendation
37. Sporny, M., Longley, D., Bernstein, G.: Data integrity ecdsa cryptosuites v1.0 (March 2025), https://www.w3.org/TR/vc-di-ecdsa/

38. Sporny, M., Longley, D., Chadwick, D., Steele, O.: Verifiable credentials data model v2.0 (Apr 2024), https://www.w3.org/TR/2024/CRD-vc-data-model-2.0-20240416/
39. UN/CEFACT: Un transparency protocol (untp) specification (2025), https://uncefact.github.io/spec-untp/, accessed: 2025-05-13
40. Wood, D., Lanthaler, M., Cyganiak, R.: RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation 25 February 2014 (2014), https://www.w3.org/TR/rdf11-concepts/
41. ZKM Contributors: Zkm architecture (2025), https://docs.zkm.io/zkm-architecture, accessed: 2025-05-13