Proving Soundness of SPARQL Query Results using Selective Disclosure of RDF Datasets and Zero-Knowledge Proofs

Christoph H.-J. Braun[✉,1, [0000−0002−5843−0316]], Jesse Wright[2,[0000−0002−5771−988X]], and Tobias Käfer[1,[0000−0003−0576−7457]]

[1] Karlsruhe Institute of Technology, Institute AIFB, Web Science
{braun,tobias.kaefer}@kit.edu
[2] University of Oxford, Department of Computer Science
jesse.wright@cs.ox.ac.uk

Abstract. Traditional SPARQL provenance has focused on explainability, while data sharing today increasingly requires privacy preservation and cryptographic verification. We thus introduce zkRDF, a data-centric approach that enables a data holder (the prover) to guarantee soundness of SPARQL query results to a data consumer (the verifier) while exposing only the minimal information needed for the proof. Unlike existing methods that prove query execution, we establish soundness of query results by proving properties about the queried RDF dataset. Given a consumer's query, the holder materializes a selectively disclosing view of the queried dataset, revealing required information and cryptographically hiding the remainder. Zero-Knowledge Proofs (ZKPs) guarantee both the integrity of the derived dataset and the adherence of hidden elements to desired constraints, such as numeric bounds. The consumer verifies the proofs and obtains the desired query results from the dataset. Importantly, proof verification ensures dataset validity and, therefore, guarantees sound query results. We present zkRDF's methodology, detail the interpretation of SPARQL queries to produce sound results, and prove soundness of our approach. We discuss zkRDF's support for SPARQL features and show that our proof-of-concept implementation outperforms an approach that proves query execution by three orders of magnitude.

Keywords: RDF · Verifiable Credentials · Zero-knowledge Proofs

1 Introduction

Sharing data via the Web is increasingly requiring cryptographic assurance of shared information – prescriptions in healthcare [26], the incoming Digital Product Passport [8], or personal information via the EUDI Wallet [17]. The twin issues of data integrity and data privacy arise: While ensuring cryptographic data integrity helps to combat fraud and creates accountability [1], compliance or regulation barriers like the EU's GDPR [22], mandate ensuring data privacy to protect e. g. citizens' private information or business secrets. The research

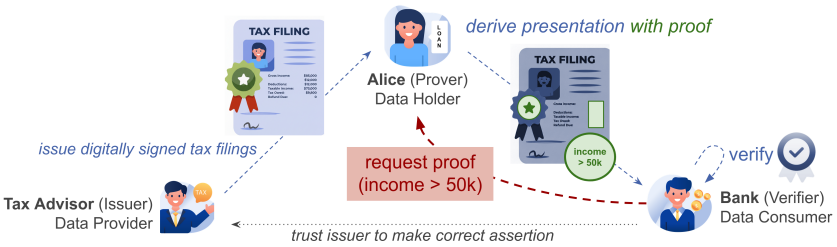

Fig. 1: The problem: Querying data to receive sound results with provenance.

question follows immediately: How can a data consumer specify which attested information they need to receive from a data holder, such that they can verify the soundness of the retrieved results – all while enabling the data holder to maintain the privacy of information unrelated to the inquiry?

Consider Figure 1: To receive a business loan, a bank requires proof that an applicant, e. g. freelance consultant Alice, has sufficiently stable income to cover the monthly payments. Instead of plainly submitting sensitive tax filings, which were prepared and digitally signed by Alice's tax advisor, Alice only needs to prove to the bank that (a) the income is above the required threshold and (b) that the income was signed off as correct by the tax advisor.

To prove such information, Europe's leading cryptographers strongly advocate for the application of Zero-Knowledge Proofs (ZKPs) on top of digital signatures [2]. Using a ZKP, a prover is able to convince a verifier that a claim is true without the verifier learning any additional information [21]. To model asserted data and cryptographic provenance information, the W3C Verifiable Credentials (VCs) [38] recommends a graph-based data model based on the Resource Description Framework (RDF) [15]. Previous efforts on Linked Data Integrity Proofs [3] were recently complemented by a definition of RDF-based semantics for selective disclosure [7]. It is logically sound to apply querying and reasoning techniques on selectively disclosing RDF datasets [7], e. g. VCs and their presentations, even when applying ZKPs. This logical consistency [7] enables the central idea of our approach: using SPARQL [24], the standard query language for RDF, to express which data or quality such as numeric bounds to disclose.

We introduce *zkRDF*, a data-centric approach to prove and verify soundness of SPARQL query results. Upon receiving a query from a data consumer, a data holder executes the received query and discloses a subset of the queried dataset with attached proofs about the desired information as expressed by the query. The data consumer receives such a presentation of the queried dataset including corresponding proofs. This is called a "selectively disclosing dataset" [7] where proof verification serves as a validity check that ensures soundness of subsequent queries. With that, the data consumer is able to obtain the desired sound query results. Our approach is data-centric: The generated proof consists of sub-proofs on information from the queried dataset. See Listings 1.1 - 1.3 for an example.

```
[] a fin:TaxReturn ;
   fin:about ex:alice ;
   fin:provider ex:trustedTaxAdvisor;
   fin:taxPeriod "2024" ;
   fin:taxableIncome
     [ fin:currency "USD" ;
       fin:value "77000"^^xsd:integer ] .
 # signature details omitted for brevity
```

Listing 1.1: Example: Simplified tax filing; original graph signed by the tax adivsor. It underlies the selectively disclosing dataset presented in Listing 1.3.

```
SELECT ?taxAdvisor                                  # disclose which tax advisor
WHERE {
    ?return2024     a     fin:TaxReturn ;
        fin:about         ?user ;                   # (hidden) user ID to link across VCs
        fin:provider      ?taxAdvisor ;             # find tax advisor
        fin:taxPeriod     "2024" ;
        fin:taxableIncome [ fin:currency "USD" ;
                            fin:value ?income2024 ] . # find income value
    FILTER(?income2024 >= 50000)                    # check income threshold
}
```

Listing 1.2: Example: Bank's query for income above 50000 (USD) in 2024. The query is sent from the data consumer (the bank) to the data holder (Alice).

```
GRAPH _:0_4 {
    _:0_0 a fin:TaxReturn .                              # this is tax return information
    _:0_0 fin:holder _:0_7 .                             # the holder's ID is hidden
    _:0_0 fin:provider ex:trustedTaxAdvisor .            # the signing tax advisor's ID
    _:0_0 fin:taxPeriod "2024" .                         # tax return from 2024
    _:0_0 fin:taxableIncome _:0_22 .                     # taxable income is attesed
    _:0_22 fin:currency "USD".                           # as USD
    _:0_22 fin:value _:0_37 .                            # but income value is hidden
}
GRAPH _:presentationProofGraph {
    _:cproof rdf:type zkp:CompositeProof .               # proof has multiple parts
    _:cproof zkp:hasComponent _:p0 , _:p1 .              # two parts in this example
    _:0_7 spok:hasSchnorrResponse "a..f"^^xsd:base64Binary.# original value is proven known
    _:p0 rdf:type bbsp16:PoKS16 .                        # proof of knowledge of signature
    _:p0 bbsp16:isProofOfKnowledgeOfSignatureOverGraph _:0_4 . # signature over tax return
    _:p1 rdf:type lg16:LegoGroth16ProofOfRangeMembership . # proof of numeric bounds
    _:p1 lg16:hasWitness _:0_37 .                        # value to be proven
    _:p1 lg16:hasLowerBound "50000"^^xsd:nonNegativeInteger .  # to be greater than 50k
    # more proof details omitted for brevity
}
```

Listing 1.3: Example: An excerpt of a selectively disclosing RDF dataset. Given a query (Listing 1.2), such a dataset is derived by the data holder from an underlying digitally signed dataset (Listing 1.1). Blank nodes serve as stand-ins for hidden terms, and knowledge of corresponding underlying terms is cryptographically proven. A proof of knowledge of signature shows that the original information had been signed by a particular tax advisor. A proof of numeric bounds shows that the originally asserted income is indeed above the threshold. This dataset is presented by the holder to the bank. The bank verifies these proofs and, using a re-written version of Listing 1.2's query (cf. Listing 1.4), derives the desired query results from the obtained dataset themself; thus cryptographically proven to be sound. For further details, see Section 4.

Our evaluation shows that zkRDF supports a core SPARQL feature fragment, sufficient for today's typical use cases of VCs. In terms of performance, we compared our proof-of-concept (PoC) implementation[3] to a prototype following an alternative but comparable approach [41]. Their approach relies on a zero-knowledge virtual machine[4] and is shown to be three orders of magnitude slower on their benchmark than our approach. We highlight our contributions:

- an initial categorization of data-centric and computation-centric approaches to proving soundness of query results (Section 3)
- zkRDF (Section 4): A conceptual framework and a proof of soundness of resulting SPARQL query results (formal foundations available in Appendix A).
- its evaluation (Section 5) along the three dimensions of
  - competence – in terms of SPARQL feature support
  - performance – to compare implementations to alternatives
  - privacy and security – to discuss potential limitations

  providing an initial evaluation framework.

We first cover the necessary foundations (Section 2), then present our contributions (Sections 3-5), and conclude with a vision for future research (Section 6).

2 Preliminaires

We briefly cover the fundamental concepts of SPARQL which the zkRDF approach uses to achieve selective disclosure of RDF datasets.

RDF [15], the Resource Description Framework, provides a graph-based data model. We re-use the common formalization of RDF datasets from [7]. We agree to the choice of semantics for RDF datasets from [7]: When merging two RDF datasets, blank nodes between the two RDF datasets must be re-labeled to avoid co-references, and new graph names must be minted for the default graphs of the respective RDF datasets.

The Verifiable Credentials (VCs) data model [38], a W3C Recommendation, models assertions and cryptographically assured provenance, e. g. digital signatures, as an RDF dataset. In that RDF dataset, the claims and credential metadata form the unnamed default graph, and a particular claim links to a named graph with corresponding cryptographic provenance information, i. e. the proof graph. Nanopublications [29] offer an alternative RDF-based data model where provenance information is modeled in a named graph and linked to a named graph of assertions.

SPARQL [24] is the W3C-recommended query language for RDF. The fundamental component of a SPARQL query is the Basic Graph Pattern (BGP), a set of triple patterns. The evaluation of a BGP against an RDF graph yields a multiset of solution mappings. Each solution mapping maps variables to RDF terms.

[3] `https://github.com/uvdsl/rdf-zkp-sparql`

[4] We acknowledge that other, more performant, options for proving properties of computations are available, e. g. using arithmetic circuits [36].

The semantics of complex SPARQL queries are formally defined by an algebra over these multisets of solution mappings. One of the key algebraic operators is `FILTER`, which reduces the multiset of solution mappings based on a boolean expression. Such expressions are constructed from variables, RDF literals, and built-in functions, including logical connectives and relational comparisons.

Selective disclosure refers to the concept of proving properties of a dataset while hiding some or all of the data [5]. One way to implement this is through zero-knowledge proofs (ZKPs) [21]: A prover is able to convince a verifier that a claim is true without the verifier learning any additional information (zero-knowledge). An example of a claim is: `My secret age` w `is greater than 21`. More formally, a claim consists of the following elements [12]:

`relation:` what is to be proven, i. e. $secret\ value > public\ value$;
`statement:` set of public inputs to the relation, i. e. `21`;
`witness:` set of private inputs, secret to the prover, e. g. `25`.

ZKPs can be used to prove a number of different relations, such as proving numeric bounds [12, 16] or set (non-)membership [40]. In the domain of VCs, the most common use is proving knowledge of signature [9, 10, 35] on credential data while simultaneously revealing some data in the credential and hiding the remainder. Proof composition, i. e. combinations of ZKPs on the same witness data, is enabled by following the commit-and-prove scheme [13, 27].

3 Related Work

Provenance surrounding SPARQL is a long researched field [19, 20, 23, 25]. We note two works on provenance of data integrity in external query execution: An early work on outsourcing verifiable BGP queries [33] compiles the algorithm for Triple Pattern Fragments (TPF) evaluation into arithmetic circuits to prove completeness and soundness of query results from a trusted data source but provided by an untrusted third party. VeriDKG [44] relies on a blockchain-based network of "auditors" to generate cryptographic proofs on the integrity of SPARQL query execution via verifiable set operations. Both works [33, 44] require a transparent infrastructure where the query-executing third party must be able to fetch and process the underlying data in plain, compromising data privacy. In contrast, our approach prioritizes data sovereignty: it requires no blockchain or public storage; SPARQL query results are selectively disclosed by the data holder directly to the data consumer (cf. Figure 1).

To our knowledge, only limited work exists at our intersection of SPARQL and ZKPs for privacy-preserving data provenance. We thus include related research on other query languages and works particular to VCs [38]. We categorize existing approaches into two classes: computation-centric (proofs of query execution) and data-centric (proofs of data properties). Our work falls into the latter category. This initial categorization highlights the fundamental differences in approach; a full formal analysis is out of scope here.

Computation-centric approaches to zero-knowledge querying focus on proving the correctness of a result that is computed or aggregated from a dataset. The primary goal is to provide a verifiable answer to an (often analytical) query without revealing the underlying data that was used in the computation.

For SPARQL, an explorative work on provable provenance and Privacy-Preserving Queries [41] uses a zero-knowledge Virtual Machine (zkVM) to prove the execution and thus results of a query. In this SPARQL-in-zkVM approach, the zkVM creates a cryptographic proof that a SPARQL query engine indeed produced particular query results, given an input SPARQL query. In their use case, the dataset includes digitally signed RDF graphs, i.e. VCs. The program executed within the zkVM also includes a verification of those VCs to prove that the SPARQL results were obtained from digitally signed RDF graphs.

For SQL, ZKSQL [30] proves correct execution of SQL queries on relational databases. This approach decomposes a query based on the standard relational algebra query plan into SQL operators like `join`, `filter`, and `aggregate`. It then relies on an interactive ZK proof protocol that requires real-time, back-and-forth communication between the prover and verifier to generate and validate a proof of correct SQL query execution.

For Cypher, ZKGraph [42] targets graph databases to prove complex algorithmic operations like pathfinding and multi-hop traversals. This approach employs an "expansion-centric" model where a query is broken down to the core primitive, i. e. graph traversal (node expansion), and other operations are treated as variations (that have certain "attributes") of this primitive. The prover generates a single, self-contained proof that the query results had been correctly computed.

Data-centric approaches to zero-knowledge querying focus on proving that a specific subset of data is authentic and satisfies a set of declarative properties. The primary goal is the selective and verifiable disclosure of data items themselves or properties of them, not a new value computed from them.

A first approach to ZKPs on RDF-based credentials [43] only allows selective disclosure of complete triples of an RDF graph. The current Data Integrity BBS Cryptosuites [4], a Candidate Recommendation Draft by the W3C Verifiable Credentials Working Group, follows a similar approach.

Considering RDF-based semantics [7] provides a more granular approach for selective disclosure: This approach allows to selectively disclose RDF terms fine-grained within a quad of an RDF dataset, e. g. to prove numeric bounds or set (non-)membership on individual RDF terms; or equality of RDF terms across quads. Moreover, in this approach, it is proven to be logically sound to query and to reason on VCs and their presentations, even when using ZKPs. Our work builds on the semantic foundations provided in [7] and extends it by enabling the definition of what to prove using SPARQL.

For querying digital credentials, the Digital Credentials Query Language (DCQL), a JSON-based language defined by the OpenID4VP standard [39], provides a credential-format-independent way for verifiers to request specific data. DCQL is deliberately limited to prevent data leakage: A verifier can only ask for entire credentials, credential formats, or attested attributes, and does

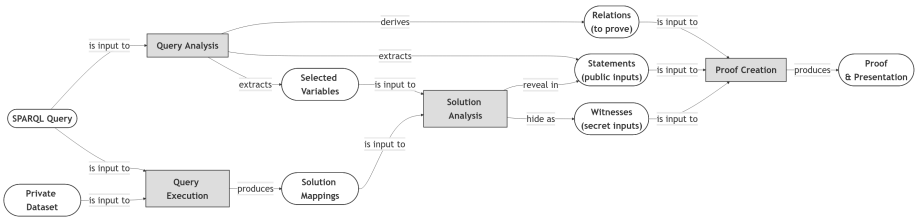

Fig. 2: An illustration of zkRDF's conceptual framework to achieve query-derived selective disclosure. Rounded nodes represent data, gray boxes represent procedures. Edge labels qualify the link between data and a procedure.

not support expressing constraints such as numeric bounds. As shown in [34], SPARQL queries can be used instead of DCQL within OpenID4VP [39] to replace JSONPath pointers, and to selectively disclose RDF triples [43]. In contrast, our approach is not limited to the use case of VCs, but it is a general approach to prove soundness of SPARQL query results from digitally signed RDF datasets.

4 The zkRDF Approach

In this section, we introduce *zkRDF*, a data-centric approach to prove and verify soundness of SPARQL query results from digitally signed RDF datasets. That is, in contrast to the computation-centric approaches, which produce a proof of execution, our approach provides proofs about the underlying dataset such that the verifier can obtain the query results.

Re-visit Figure 1 for an example: The data consumer sends a query to the data holder. The core challenge is not just deriving a view of an existing digitally signed dataset, but preserving cryptographic integrity as asserted by the data provider via their signature. The data consumer only trusts the data provider to sign correct assertions. Therefore, when the holder is to reveal some dataset terms to the consumer and hide others, the holder must generate cryptographic proof that the disclosed and hidden terms were originally signed by the provider. This requires proving knowledge of signature, equality constraints (that occurrences of a blank node map to the same underlying hidden value) and numeric bounds.

4.1 Conceptual Framework

zkRDF systematically transforms a SPARQL query and its solution into a set of relations, statements, and witnesses to be proven. This approach, illustrated in Figure 2, consists of the following abstract procedures:

Query Execution. Evaluate the SPARQL query to obtain solution mappings.
Query and Solution Analysis. Derive relations from graph patterns and expressions; determine which RDF terms are public and which remain private.
Proof Creation. Create a proof over the relations, statements, and witnesses.

We present zkRDF's design, a direct extension of [7] to realize the abstract procedures, yielding a conceptual framework that guides concrete implementations.

On the data holder's (prover's) side. Having received a SPARQL query (cf. Listing 1.2), a data holder is to derive a selectively disclosing dataset (cf. Listing 1.3). When evaluating the query over the original dataset (cf. Listing 1.1), the holder must determine for each solution mapping

(a) which occurrences of RDF terms in the original dataset should be revealed,
(b) which occurrences RDF terms in the original dataset should remain hidden,
(c) what needs to be proven to assure soundness of the query results.

To this end, zkRDF traces the "origins" of each solution mapping, i. e., tracing which occurrences of which terms to produced a particular solution mapping. This may seem similar to a *provenance semiring* [23]: In lineage provenance, merging mappings implies a join such that tuples `t1` *and* `t2` are required for the solution mapping to exist. But semantics and purpose are different in zkRDF: On a term-level rather than tuple-level, zkRDF collects all indices of term occurrences where the occurrences are relevant to the current solution set (cf. definition of $\texttt{merge}^+$ for set union; Appendix A) to cryptographically prove Equality Constraints. If an occurrences is not collected in zkRDF, the solution mapping still exists – that occurrence is just not proven to be the same as the others.

The prover's algorithm, formally outlined in Algorithm 1 with its formal underpinnings available in Appendix A, can intuitively be understood as:

Query Execution: Evaluate the query on the original dataset up to projection.
Query and Solution Analysis: For each solution mapping:

1. Using the set of projected variables (V_{proj}), determine which terms should be revealed (in proof statements) and which hidden (as witnesses):
   → *(a)* Terms mapped to variables to be projected are to be revealed. Constants in the query are revealed; they were already known by the verifier.
   → *(b)* All other mapped terms are to be hidden.
2. For each mapped RDF term, identify its occurrences in the dataset that are relevant to the current solution mapping (cf. definition of $\texttt{eval}^+$ in Appendix A for formal details). This yields the "origin trace".
   → *(a)* Occurrences of revealed terms will be used in relations' statements.
   → *(b)* Occurrences of hidden terms will be used as relations' witnesses.
3. Based on the "origin trace", the query and its solution, identify the witnesses, statements, and relations that are required to be proven:
   Signature Proofs to cryptographically prove that revealed and also hidden terms had originally been signed by the data provider.
   → *(c)* For each graph of the original dataset that contributes a term occurrence to the "origin trace", an instance of this proof is required.
   Numeric Bounds to cryptographically prove that the query's `FILTER` expressions indeed hold on a corresponding hidden term.
   → *(c)* For each numeric bounds `FILTER` expression with a non-projected variable, an instance of this proof is required.
   Equality Constraints to cryptographically prove equivalence across different occurrences of the same term when that term remains hidden.

Algorithm 1 zkRDF Prover (Data Holder) Algorithm | Definitions available in Appendix A

Require: Original dataset D, SPARQL Query Q, Set of Projected variables V_{proj} (Step *1.*)
Ensure: Presentation P

```
Ω⁺ ← eval⁺(Q, D)                          ▷ Evaluate query with occurrence tracing (includes Step 2.)
P_spec ← ∅                                 ▷ Initialize set of proof specifications
C_set ← ∅                                  ▷ Initialize set of equality constraints
for all μ⁺ ∈ Ω⁺ do                         ▷ Analyze each solution mapping (Step 3.)
    N_μ⁺ ← {n | graph n contains term mapped in μ⁺}
    for all n ∈ N_μ⁺ do                    ▷ PoKS: Signature Proofs
        I_w ← ∅
        I_s ← ∅
        for all ?v ∈ dom(μ⁺) do
            if ?v ∉ V_proj then I_w ← I_w ∪ μ⁺(?v).I      ▷ Collecting occurrences to hide
            end if
            if ?v ∈ V_proj then I_s ← I_s ∪ μ⁺(?v).I      ▷ Collecting occurrences to reveal
            end if
        end for
        w ← {σ_w, I_w}                     ▷ Witness: signature and occurrences to hide
        s ← {ρ_σ, I_s}                     ▷ Statement: issuer's public key and occurrences to reveal
        P_spec ← P_spec ∪ {(PoKS, s, w)}
    end for
    for all Expression E ∈ E_bounds where ?v ∉ V_proj do   ▷ PoNB: Numeric Bounds
        w ← {μ⁺(?v).T}                     ▷ Witness: secret numeric value
        s ← {(min, max)}                   ▷ Statement: bounds from FILTER
        P_spec ← P_spec ∪ {(PoNB, s, w)}
    end for
    for all ?v ∈ dom(μ⁺) \ V_proj do       ▷ Equality Constraints
        if |μ⁺(?v).I| > 1 then
            C_set ← C_set ∪ {μ⁺(?v).I}
        end if
    end for
end for
π ← CreateProofs(P_spec, C_set)            ▷ Handled by the low-level proof system
D' ← CreateSelectivelyDisclosingDataset(D, Ω⁺, V_proj, π)   ▷ Serialization of Presentation
return P = D'
```

→ *(c)* For each hidden term with at least two occurrences in the "origin trace", an equality constraint is required.

Proof Creation: Create a corresponding proof of all identified relations and equality constraints given the public statements and secret witnesses. Based on the original dataset, the relations, statements, witnesses, and the proof, create a corresponding dataset to be presented to the data consumer for verification, i. e. a selectively disclosing dataset (cf. Listing 1.3). Each occurrence of a hidden RDF term is replaced with a stand-in blank node; occurrences cryptographically proven to be equivalent are assigned the same blank node. Note that it is logically sound to hide the particular bound term using a stand-in blank node [7]. The data holder can thus prove that the particular term was indeed signed by the data provider, i. e. the tax advisor, despite the term remaining hidden.

On the data consumer's (verifier's) side. The data consumer receives the selectively disclosing dataset from the data holder. To obtain the desired query results, the consumer's algorithm[5] is formally outlined in Algorithm 2:

1. Cryptographically verify the proofs provided in the received dataset

[5] No new algorithms are required besides the straight-forward re-writing of the query.

Algorithm 2 zkRDF Verifier (Data Consumer) Algorithm

Require: SPARQL Query Q, Presentation P, Projected variables V_{proj}
Ensure: SPARQL query result Ω or **Reject**
  if $\neg$VerifyProofs(P) **then** ▷ Lemma 1: Cryptographic Soundness
    return Reject ▷ Proof is invalid or missing
  end if
  $Q_{rewritten} \leftarrow Q$
  for all Expression $E \in E_{bounds}$ in $Q_{rewritten}$ **do** ▷ Theorem 2: Query Rewriting
    if variable $?v$ in $E \notin V_{proj}$ **then**
      $P_R \leftarrow$ MapToGraphPattern(E) ▷ Derive corresponding proof graph pattern
      Replace `FILTER` expression E with P_R in $Q_{rewritten}$
    end if
  end for
  $\Omega \leftarrow$ `eval`($Q_{rewritten}, P$) ▷ Evaluate modified query on verified dataset
  return Ω

```
SELECT ?taxAdvisor                                    # disclose which tax advisor
WHERE {
    ?return2024     a     fin:TaxReturn ;
        fin:about         ?user ;                     # (hidden) user ID to link across VCs
        fin:provider      ?taxAdvisor ;               # find tax advisor
        fin:taxPeriod     "2024" ;
        fin:taxableIncome [ fin:currency "USD" ;
                            fin:value ?income2024 ] . # find income value
    # substitute FILTER(?income2024 >= 50000)         # check income threshold
    ?f0 rdf:type lg16:LegoGroth16ProofOfRangeMembership . # proof of numeric bounds
    ?f0 lg16:hasWitness ?income2024 .                   # the secret income
    ?f0 lg16:hasLowerBound 50000 .                      # required threshold
    ?f0 lg16:hasUpperBound "18446744073709551615"^^xsd:nonNegativeInteger . # MAX u64
    # more proof details omitted for brevity
}
```

Listing 1.4: Example: Re-written query from Listing 1.2 substituting `FILTER(?income2024 > 50000)` with a graph pattern by directly mapping values.

2. Evaluate the SPARQL query on the received dataset as follows:
   In case of a SPARQL query that
   - does not have a `FILTER` for numeric bounds, evaluate the query directly.
   - includes `FILTER` expressions exclusively over projected variables, evaluate the query directly.
   - includes `FILTER` expressions also over non-projected variables, the verifier cannot evaluate these `FILTER`s directly (relevant terms are hidden). The query is thus rewritten: the `FILTER` is replaced by a graph pattern that specifies a corresponding proof of numeric bounds to be present in the proof of the selectively disclosing dataset. Recall that verifying this proof, i. e. that the `FILTER` was correctly applied, is done in Step 1.

We illustrate the intuitive equivalence between a `FILTER` expression of numeric bounds and proof relation of numeric bounds. Let there exist a variable binding for `?income2024` from the original query (Listing 1.2) to `"77000"^^xsd:integer` from the original dataset (Listing 1.1). The RDF literal remains hidden under blank node `_:0_37` in the selectively disclosing dataset (line 8 of Listing 1.3). For the data consumer to obtain the desired query results from the corresponding selectively disclosing dataset (Listing 1.3), the `FILTER` statement of the original query (Listing 1.2) must be re-written (cf. Listing 1.4):

As more formally outlined in Appendix A, the `FILTER` defines `?income2024` $\in$ $[\texttt{50000}, \infty)$, where ∞ refers to the maximum value supported by the proof system. Therefore, a corresponding proof of numeric bounds is required to be provided by the holder in the selectively disclosing dataset. Specifics of the proof and its modeling are dependent on the proof system employed, but the construction of the corresponding graph pattern is a straight-forward mapping of values: In our example Listings 1.3 and 1.4, we use LegoGroth16 [12] as modeled in [7]. Lower and upper bounds are directly mapped from the `FILTER` expression. The witness is the non-projected variable. If such a graph pattern is found in the selectively disclosing dataset and all its proofs are valid, then the data consumer may accept the hidden value to be indeed within the specified numeric bounds.

The proposed rewriting of `FILTER` expressions into graph patterns, e. g. numeric bounds and associated proofs, aligns with the broader principle of translating query constraints into declarative forms; a technique exemplified by SPARQL-to-SHACL [28]. By embedding ZKPs directly into the RDF graph structure, we transform ephemeral query filters into verifiable cryptographic assertions.

4.2 Soundness of SPARQL Query Results

This approach is sound and generalizable, following the reasoning presented in [7]: (a) the selective disclosure of RDF terms corresponds to simple entailment, and (b) proof verification serves as a validity check on the selectively disclosed dataset. Thus, provided the data consumer trusts the issuer's attestation, successful proof verification allows the consumer to accept the selectively disclosed dataset as true.

We build on this reasoning to demonstrate that additional proofs regarding the quality of hidden RDF terms, e. g. such as our proof of numeric bounds, are equally sound. For a system to prove a "true" quality of a cryptographically asserted value while preserving privacy, the following properties must interlock:

1. **Origin** (`unforgeability` & `membership`): The digital signature guarantees unforgeability, meaning the value ν is a valid member of the set of values attested by the issuer (which are assumed to be true).
2. **Commitment** (`binding` & `hiding`): The binding property ensures the prover cannot swap the signed value ν for a different value ν' during proof generation. The hiding property conceals the actual value ν computationally.
3. **Verification** (`soundness` & `zero-knowledge`): The proof system directly enforces the soundness property, guaranteeing that a statement cannot be proven if it is false. The zero-knowledge property ensures that the verification process leaks no information about ν beyond the truth of the statement.

Any successful proof of a false predicate would imply a mathematical collision in the underlying cryptography. Consequently, proven qualities, e. g. numeric bounds or set (non-)membership, must hold true for the bound, authentic values.

We establish the soundness of our approach by composition: first, the validity of the cryptographic proof (Lemma 1), and second, the semantic preservation of the query rewriting (Theorem 1). See Appendix A for formal foundations.

Lemma 1 (Cryptographic Soundness). *Let π be a ZKP generated for relation $R(\nu)$ on committed value ν. For any proof π and its relation $R(\nu)$: proof π verifies successfully if and only if $R(\nu)$ holds (except with negligible probability).*

This follows directly from the binding property of the commitment and the soundness property of commit-and-prove systems. These properties are established through universal composability [13] and the formal analysis of Sigma protocols [11] and commit-and-prove SNARKs [31].

Theorem 1 (Soundness of Query Rewriting). *Rewriting a `FILTER` expression E to graph pattern P_R of a proof of relation R preserves query semantics.*

Proof. Let Ω denote a multiset of solution mappings μ. Recall that a `FILTER` $F(E, \Omega)$ retains a solution mapping μ if and only if $E(\mu)$ is `true`. Consider μ a solution mapping produced by the original query on the original dataset, and let $\mu(?v) = \nu$ denote a variable binding from variable $?v$ to value ν in that dataset.

- *Witnesses & Entailment:* In the derived selectively disclosing dataset, secret value ν remains hidden, replaced by a stand-in blank node n. As per Theorem 1 of [7], this is equivalent to simple entailment and logically sound.
- *Rewriting the query:* `FILTER` F is replaced with graph pattern P_R that represents a proof of relation R. By construction, relation R is logically equivalent to the filter expression E, i. e. $R(\nu) \iff E(\mu)$.
- *Evaluating the re-written query on the selectively disclosing dataset:* The graph pattern P_R matches n and a corresponding proof π, yielding solution mapping μ' with $\mu'(?v) = n$. By Lemma 1 and Theorem 1 of [7], the proof π is valid if and only if the underlying value ν satisfies the relation as encoded in the triples matching P_R.

It follows: Evaluation of graph pattern P_R on the selectively disclosing dataset produces μ' if and only if $E(\mu)$ is `true` in the original dataset. □

Given Lemma 1 and Theorem 1, the query rewriting approach is sound: query results over the selectively disclosed dataset with verified proofs contain only solution mappings that would be valid over the original dataset.

5 Evaluation

We evaluate our zkRDF approach by assessing its competence in terms of which fragment of SPARQL [24] features are or could be supported. In addition, we compare performance between our proof-of-concept implementation[6] with the SPARQL-in-zkVM project [41]. Finally, we discuss security and privacy implications in this approach. We consider the evaluation along the three dimensions of competence, performance (or efficiency), and security & privacy a potential framework to aid other researchers in evaluating other comparable approaches.

Table 1: Currently considered fragment of SPARQL features in zkRDF; feature support (✓), partial support (⊃), theoretic support (△), no support (×).

Feature	Support	Notes
`SELECT`	✓	returns selectively disclosing dataset
`ASK`	⊃	only `true` results provable
`CONSTRUCT`	✓	treated as `SELECT` with full projection
`DESCRIBE`	×	ambiguous semantics
`JOIN`|`UNION`|`OPTIONAL`	✓	proven via matches in selectively disclosing dataset
`MINUS`	×	cannot prove non-existence
`VALUES`	✓	value-constraints on variables (as usual in SPARQL)
`BIND`	⊃	only static assignment of terms supported
`FILTER` (numeric)	⊃	integers supported; limited coercion, e. g. DateTime
`FILTER` (string)	×	not provable in ZK with current RDF term encoding
`IN`, `NOT IN`	△	possible via set membership proofs (future work)
`GROUP BY`, `DISTINCT`	✓	ZK not necessary; proven by inspection
`ORDER BY`	×	not provable in ZK with current RDF term encoding
`OFFSET`	×	not provable in ZK with current RDF term encoding
`LIMIT`	✓	proven by cardinality of final query results
`Aggregates`	△	possible via arithmetic circuits (future work)
`Property Paths`	△	possible via back-tracing (future work)

5.1 Competence: Supported Fragment of SPARQL Features

We demonstrate that our data-centric approach provides a viable foundation for proving the soundness of query results. Our evaluation focuses thus on a core SPARQL fragment. The supported features already enable today's use cases of VCs where `ORDER BY`, `OFFSET`, or string operations are typically not required.

We summarize the currently considered fragment of SPARQL features in Table 1. As a query is answered using a selectively disclosing dataset, proving existence of graph patterns is trivial. Numeric filter expressions[7] are limited by the applied cryptography to integers; other data types (e. g. DateTime) are mapped to integers. A literal's data type can be revealed independently from its value; no ambiguity is introduced. `VALUES` may be used to express value-constraints on variables as usual. We highlight limitations and future work:

– Fundamentally, we cannot prove non-existence of pattern matches or solution mappings. Our approach hinges on proving knowledge of signature (the provider's) on revealed/hidden data. Only if the holder has a signature of a provider on some data, the holder can create proofs about that data. If the holder does not have a signature on some data, they are unable to cre-

[6] `https://github.com/uvdsl/rdf-zkp-sparql`

[7] In our implementation of the concept, we currently only support "simple" relational expressions where only one variable exists. This is not an assumption made by the zkRDF approach, but merely a limitation of the current implementation.

ate a proof about that data. Therefore, the holder cannot produce a proof that a provider did not sign certain data. Even if the holder proves that a signed dataset does not contain certain data, it does not mean that the holder does not have a separate dataset where that data (potentially with signature) exists. We thus cannot prove application of `MINUS` or prove `false` for `ASK`.
- `BIND` is supported for static assignment of terms. Advanced use of expressions to create e. g. numeric literals are currently not supported. We highlight that bound terms that do not exist in the dataset may result in solution mappings that could not have existed without those externally provided terms. zkRDF proves the data underlying those solution mappings, such that the consumer obtains these mappings using bound values on top of the proven data.
- With the current RDF term encoding for ZKP application (cf. Sec. 3.2 of [7]),
  - we cannot prove an ordering between hidden solution mappings. We thus cannot prove application of `OFFSET` or `ORDER BY`.
  - we treat a string literal value as an opaque unit and cannot operate on the characters of a string. We thus cannot prove application of operations on a string (length, character at index, etc.) in a `FILTER` expression.

  More potent term encoding thus remain important future work.
- With the current proof system, capabilities could be extended by future work
  - on proofs of set (non-)membership proofs for `IN`/`NOT IN`.
  - on custom arithmetic circuits to proof application of `Aggregates`.
  - on back-tracing `Property Paths` in a solution mapping.

5.2 Performance: PoC Implementation

The work on proving the execution of a SPARQL query in a zkVM (SPARQL-in-zkVM) [41] provides a small testing benchmark, which we refer to as `risc0-bench`. We emphasize that this benchmark is not our contribution.

The `risc0-bench` testing benchmark provides four VCs under `Ed25519` Linked Data Signatures with varying yet overlapping contents from different issuers. Three queries are provided:
- `can-drive`: a query with 3 `FILTER` statements expressing numeric bounds.
- `employment-status`: a simple graph pattern query.
- `show-all`: a generic "select all" `{ ?s ?p ?o }` graph pattern query.

The queries do not include aggregation functions and are instead focused on selective disclosure of personal data. This allows us to compare the computation-centric SPARQL-in-zkVM with our data-centric zkRDF.

We acknowledge: The benchmark is indeed small, but it accurately reflects the scale of today's credentials, which are small, document-centric graphs. Developing a comprehensive, large-scale benchmark for zero-knowledge SPARQL evaluation thus remains an important future work.

Performance Comparison. We summarize our benchmarking results in Table 2. We ran the release version of `risc0-bench` for the SPARQL in zkVM approach on a consumer laptop (AMD Ryzen 7 PRO 5850U): Proving the execution of the respective queries using a zkVM thus takes approximately 26 minutes (`can-drive`), 24 minutes (`employment-status`), and 29 minutes (`show-all`).

Table 2: Performance comparison in milliseconds (ms).

	SPARQL-in-zkVM		zkRDF	
Task	**Prove**	**Verify**	**Prove**	**Verify**
can-drive	1,571,816.15	7,792.54	141.61	27.63
employment-status	1,463,066.24	6,782.48	5.51	5.38
show-all	1,736,362.18	8,878.46	437.02	1,650.80

Table 3: Proving Query Performance Metrics in milliseconds (ms).

Query	Preparation	Specify Proof	Create Proof	Present Proof	**Total**
can-drive	1.59	1.36	137.74	0.90	141.80
employ-status	0.57	0.38	4.22	0.23	5.45
show-all	9.96	30.88	375.78	18.60	435.30

Our implementation of zkRDF, based on the work provided in [7], uses the `BBS+` signature scheme instead of `Ed25519`: We therefore created `BBS+` signatures for the credential data provided by `risc0-bench`. The modified version of `risc0-bench`[8] results in the performance numbers shown in Table 2; collected by `criterion.rs`. zkRDF achieves a 3 orders of magnitude improvement in proof generation time compared to SPARQL-in-zkVM.

Scalability. The `show-all` query exhibits a total verification time of 1650 ms, though this is primarily due to roughly 1300 ms of unoptimized internal data handling when extracting and deserializing cryptographic proof information. The core cryptographic verification requires only 350 ms, which is still significantly longer than the 5 ms for `employment-status` due to `employment-status` verifying only a single solution mapping, while `show-all` must verify 85.

Proof and verification complexity scales as indicated in Section 4.1: per solution mapping $\mu \in \Omega$, there exists (a) a knowledge of signature proof for each graph relevant to μ and (b) a numeric bounds proof for each applicable `FILTER`, scaling as $\sum_{\mu \in \Omega}(|G_\mu| + |F_\mu|)$, where Ω is the set of solution mappings, G_μ the graphs relevant to μ, and F_μ the applicable filters.

Performance Breakdown per Procedures. To further break down performance of the internal procedures from the conceptual framework (cf. Section 4.1), we tracked execution times manually within our code, presented in Table 3. The total proving time may thus vary slightly compared to Table 2. The `Preparation` task is comprised of the Query Execution and parts of the Solution and Query Analysis procedures. The remaining parts of the Analysis procedures are captured by `Specify Proof`, which is the "glue" between the analysis procedures and the proof system. Finally, `Create Proof` refers to the generation of the cryptographic proof and `Present Proof` refers to the serialisation of the selectively disclosing dataset which together correspond to the Proof Creation procedure.

[8] In: `https://github.com/uvdsl/rdf-zkp-sparql/tree/main/benches`

Cryptographic proof creation takes most of the overall execution time. Executing the query, deriving which proofs to create and presenting the result offer less optimization potential compared to choosing an efficient proof system.

5.3 Privacy & Security: Discussion

Information Leakage. While zkRDF significantly enhances data privacy by minimizing disclosure, it is not fully zero-knowledge in the strict cryptographic sense. Although zkRDF employs zero-knowledge proofs (ZKPs), the system as a whole may still leak metadata. For example, the number of proofs of knowledge of signatures indicate how many graphs from the underlying private dataset were used to create a solution mapping. Moreover by these proofs of knowledge of signature, the size of the respective RDF graphs is being revealed. This means that certain information may be revealed, even though ZKPs, which are indeed zero-knowledge, are being applied.

Malicious Query. Being able to prove properties about an RDF dataset in zero-knowledge does not imply immunity against information over-exposure. Consider the `show-all` query from the performance evaluation. All data from the underlying dataset is being exposed, and in addition it is proven to be signed by the corresponding issuers. To protect against such malicious queries, system architects may choose to restrict the set of acceptable queries. This can be done by directly defining a set of permissible queries in a general query language or choosing a query language with restricted expressivity, a domain-specific language like DCQL [39] (cf. Section 2). We recommend formally verifying the system's security and privacy guarantees in the spirit of and using techniques from [6, 37].

6 Conclusion

We presented zkRDF, a data-centric approach to selectively disclose, prove and verify SPARQL query results from RDF datasets. With zkRDF, we introduce the capability to prove properties of RDF-modeled data – using SPARQL as generic query interface and using ZKPs to preserve privacy where possible.

Using SPARQL is a natural extension of RDF-based semantics for selective disclosure [7]. In contrast, DCQL [39] from the VC domain exhibits deliberately limited expressivity to prevent data oversharing. While this protects users from overly eager queries, it also restricts expressing more complex properties. In the eIDAS regulation [18] for instance, qualified trust service providers (qTSP) of electronic attestation of attributes (EAA) may benefit from SPARQL's expressiveness to offer more nuanced and verifiable services. In use cases such as illustrated in Figure 1, this expressivity is not just beneficial but required.

With zkRDF, we contribute to the emerging intersection of Semantic Web technologies and privacy-enhancing technologies. As data sharing via the Web increasingly requires cryptographic assurance and privacy guarantees, we envision a future where RDF as a data model, SPARQL as the query language and ZKPs as a means of privacy-preserving provenance form the foundation to foster transparency and privacy, accountability, and – ultimately – trust on the Web.

Use of Generative AI. GPT-5 was used only for language editing, specifically to rephrase or refine text written by the authors to improve clarity and grammar. The tool was not used to generate ideas, analyses, data, figures, or any other original research content. All AI-assisted edits were manually reviewed and approved by the authors, who take full responsibility for the final manuscript.

Supplemental Material Statement: Source code and related resources are available at `https://github.com/uvdsl/rdf-zkp-sparql`.

A Formal Foundations for zkRDF

We re-use from [7]: Let $\mathcal{U}$ denote the set of HTTP URIs, $\mathcal{B}$ the set of blank nodes, and $\mathcal{L}$ the set of literals. Let $\mathcal{G}$ denote the set of RDF graphs. An RDF graph $G \in \mathcal{G}$ is a set of triples. Let triple t be $t \in (\mathcal{U} \cup \mathcal{B}) \times \mathcal{U} \times (\mathcal{U} \cup \mathcal{B} \cup \mathcal{L})$. Let $\mathcal{D}$ denote the set of RDF datasets. An RDF dataset $D \in \mathcal{D}$ is a set of named graphs [14] (n, G_n) and an unnamed *default graph* $(\_, G)$. Let graph name n be $n \in (\mathcal{U} \cup \mathcal{B})$. A triple of a named graph G_n is a quad q with $q \in (\mathcal{U} \cup \mathcal{B}) \times \mathcal{U} \times (\mathcal{U} \cup \mathcal{B} \cup \mathcal{L}) \times \{n\}$.

Let $\mathcal{V}$ denote the set of all variables, disjoint from $\mathcal{U}$, $\mathcal{B}$, and $\mathcal{L}$. A Basic Graph Pattern (BGP) is a set of triple patterns. A triple pattern tp is defined as $tp \in (\mathcal{U} \cup \mathcal{B} \cup \mathcal{V}) \times (\mathcal{U} \cup \mathcal{V}) \times (\mathcal{U} \cup \mathcal{B} \cup \mathcal{L} \cup \mathcal{V})$. The evaluation [24] of a graph pattern against an RDF graph yields a multiset Ω of solution mappings. A solution mapping $\mu \in \Omega$ is a partial function $\mu : \mathcal{V} \to (\mathcal{U} \cup \mathcal{B} \cup \mathcal{L})$. When using the `GRAPH` clause, each resulting solution mapping has a binding of n to the name of the graph that produced the solution. An expression E is a formula that can be evaluated on a solution mapping μ to yield an RDF term or an error. The `FILTER` operator evaluates an expression E for each solution mapping $\mu \in \Omega$ and retains a mapping if and only if the effective boolean value is `true`. Formally, $\texttt{eval}(E, \mu)$ being the evaluation of E for a mapping μ: $\texttt{FILTER}(\Omega, E) = \{\mu \mid \mu \in \Omega \wedge \texttt{eval}(E, \mu)\}$.

Canonicalised RDF Graphs and Datasets. Canonicalisation [32] provides an ordering to a graph's triples. Let a canonicalised RDF graph $G^+ \in \mathcal{G}$ be a list of triples, where a triple t_i is at index i; i. e. $t_i = G^+[i]$. We apply the same pattern to terms in a triple (or a quad), e. g. the subject s of a triple t_0 is $s = t_0[0] = G^+[0][0]$ such that its index tuple is $(\texttt{triple\_index}, \texttt{position\_index}) = (0, 0)$.

For zkRDF, we use a hybrid version of a canonicalised RDF dataset: All its graphs are canonicalised "on their own"; the dataset only contains these canonicalised graph without being canonicalised itself. This facilitates tracking which term occurred in which particular canonicalised RDF graph; let an *index tuple* be $I \in ((\mathcal{U} \cup \mathcal{B}) \times \mathbb{N} \times \mathbb{N})$, i. e. (`graph_name`, `triple_index`, `position_index`).

Extending SPARQL Query Evaluation with Occurrence Tracing. We extend the evaluation of a SPARQL query to directly obtain the index tuple of a variable-mapped RDF term. Let $\texttt{eval}^+$ be the indexed evaluation function which evaluates a SPARQL algebraic expression against a canonicalised, thus indexable, graph. Result of this evaluation is a multiset Ω^+ of indexed solution mappings.

An *indexed solution mapping* μ^+ extends a solution mapping to include a set of index tuples for each variable binding: $\mu^+ : \mathcal{V} \to (\mathcal{U} \cup \mathcal{B} \cup \mathcal{L}) \times \mathcal{P}((\mathcal{U} \cup \mathcal{B}) \times \mathbb{N} \times \mathbb{N})$. Each binding in μ^+ for a variable $?v$ is a pair (T, I), where $T \in (\mathcal{U} \cup \mathcal{B} \cup \mathcal{L})$ is a term and $I \in \mathcal{P}((\mathcal{U} \cup \mathcal{B}) \times \mathbb{N} \times \mathbb{N})$ is a set of index tuples where binding occurred. For a solution mapping $\mu^+(?v) = (T, I)$, we use the dot-notation to access its members, i. e. $\mu^+(?v) = (\, \mu^+(?v).T \,,\, \mu^+(?v).I \,)$.

The semantics of $\texttt{eval}^+$ are realized by indexed algebraic operators. Two indexed solution mappings, μ_1^+ and μ_2^+, are compatible if, for every variable $?v$ in the intersection of their domains, they map it to the same RDF term: $\texttt{compat}^+(\mu_1^+, \mu_2^+) \iff \forall ?v \in (\mathrm{dom}(\mu_1^+) \cap \mathrm{dom}(\mu_2^+)) \,,\, \mu_1^+(?v).T = \mu_2^+(?v).T$

$\texttt{merge}^+(\mu_1^+, \mu_2^+)$ produces a mapping μ'^+ where for any variable $?v$:

$$\mu'^+(?v) = \begin{cases} (\mu_1^+(?v).T\,,\ \mu_1^+(?v).I \cup \mu_2^+(?v).I) & \text{if } ?v \in \text{dom}(\mu_1^+) \cap \text{dom}(\mu_2^+) \\ \mu_1^+(?v) & \text{if } ?v \in \text{dom}(\mu_1^+) \setminus \text{dom}(\mu_2^+) \\ \mu_2^+(?v) & \text{if } ?v \in \text{dom}(\mu_2^+) \setminus \text{dom}(\mu_1^+) \end{cases}$$

An indexed join of two indexed solution sets is the multiset of merged mappings from compatible pairs. Other graph pattern operations are defined similarly: $\texttt{JOIN}^+(\Omega_1^+, \Omega_2^+) = \{\text{merge}^+(\mu_1^+, \mu_2^+) \mid \mu_1^+ \in \Omega_1^+ \wedge \mu_2^+ \in \Omega_2^+ \wedge \texttt{compat}^+(\mu_1^+, \mu_2^+)\}$. The rest of the SPARQL semantics remain; e. g. an indexed filter regularly applies an expression E to Ω^+: $\texttt{FILTER}^+(\Omega^+, E) = \{\mu^+ \mid \mu^+ \in \Omega^+ \wedge \texttt{eval}(E, \mu^+)\}$.

Deriving Relations, Statements, and Witnesses. SPARQL query components, i. e. graph pattern P, expressions E and projected variables V_{proj}, and pre-projection solution set Ω^+ provide all information to specify proofs about the queried RDF dataset. Let R denote a *relation* that a proof system is able to prove. Let s denote a corresponding *statement*, the set of public inputs. Let w denote a corresponding *witness*, the set of secret inputs. A *proof specification* is the tuple $(R, s, w) \in P_{spec}$. In the scope of this paper, we consider proofs of knowledge of signature (PoKS) and proofs of numeric bounds (PoNB), i. e. $R \in \{\text{PoKS}, \text{PoNB}\}$.

A PoKS proves knowledge of a valid signature. For each $\mu^+ \in \Omega^+$, let $N_{\mu^+} = \{n \mid \exists (T, I) \in \text{range}(\mu^+)\ \texttt{s.t.} \exists (n, \_, \_) \in I\}$ denote the set of graphs containing an occurrence of a term mapped in μ^+. For each graph name $n \in N_{\mu^+}$, derive a corresponding proof specification (PoKS, s, w). Witness w includes all variable-mapped occurrences of a non-projected term of graph n alongside the signature σ_w. Formally, let the witness w for a PoKS on graph n be $w = \{\sigma_w, I_w\}$ where $I_w = \{(j, k) \mid \exists\, ?v \notin V_{proj}$ s.t. $(n, j, k) \in \mu^+(?v).I\}$. The statement includes all variable-mapped projected occurrences of a term alongside other parameters ρ_σ like the issuer's public verification key. The public statement s is defined $s = \{\rho_\sigma, I_s\}$ where $I_s = \{(j, k) \mid \exists\, ?v \in V_{proj}$ s.t. $(n, j, k) \in \mu^+(?v).I\}$.

A PoNB proves that a hidden numeric value satisfies a relational constraint specified in a `FILTER` clause, for each solution mapping. Let $E_{bounds} \subseteq E$ be the subset of expressions of the form $(?v\ \texttt{op}\ C)$, where `op` is a relational operator (e. g., `>`, `<`) and C is a constant numeric literal. For each solution mapping $\mu^+ \in \Omega^+$ and for each expression $(?v\ \text{op}\ C) \in E_{bounds}$ where the variable $?v \notin V_{proj}$, a corresponding proof specification (PoNB, s, w) is derived. Witness w is the secret value: $w = \{\mu^+(?v).T\}$. Statement s contains the public relation parameters: $s = \{(\min, \max)\} = \mathbf{\{}\{(C, +\infty)\}$ if $\texttt{op} \in \{\texttt{>}\}$ **or** $\{(-\infty, C)\}$ if $\texttt{op} \in \{\texttt{<}\}\mathbf{\}}$. Let $+\infty$ and $-\infty$ denote a system's maximal and minimal supported values.

We specify *equality constraints* to prove equality of hidden RDF terms – within a graph pattern or between a node in a graph pattern and its numeric bounds. In a particular solution mapping, each occurrence of a non-projected variable-mapped RDF term needs to be proven equal. Across solution mappings, occurrence of the same term is *not proven*. Let C_{set} be the set of equality constraints derived from a single solution mapping μ^+; defined as: $C_{set} = \{\mu^+(?v).I \mid ?v \in \text{dom}(\mu^+) \wedge ?v \notin V_{proj} \wedge |\mu^+(?v).I| > 1\}$.

The set of proof specifications P_{spec} and the set of equality constraints C_{set} are provided to the proof systems to create proof π and a presentation (cf. [7]).

References

1. Balancing data sharing and privacy to enhance integrity and trust in government programs. Tech. rep., National Academy of Public Administration, Washington, D.C. (mar 2025), a research paper for the Program Integrity Alliance
2. Baum, C., Blazy, O., Camenisch, J., Hoepman, J.H., Lee, E., Lehmann, A., Lysyanskaya, A., Mayrhofer, R., Montgomery, H., Nguyen, N.K., Preneel, B., abhi shelat, Slamanig, D., Tessaro, S., Thomsen, S.E., Troncoso, C.: Cryptographers' Feedback on the EU Digital Identity's ARF (jun 2024), https://github.com/user-attachments/files/15904122/cryptographers-feedback.pdf, see also https://github.com/eu-digital-identity-wallet/eudi-doc-architecture-and-reference-framework/issues/200
3. Bernstein, G., Sporny, M.: Data integrity bbs cryptosuites v1.0. W3c candidate recommendation draft, W3C Verifiable Credentials Working Group (2024), https://www.w3.org/TR/vc-di-bbs/
4. Bernstein, G., Sporny, M.: Data Integrity BBS Cryptosuites v1.0: Achieving Unlinkable Data Integrity with Pairing-based Cryptography. Candidate recommendation draft, World Wide Web Consortium (W3C) (Apr 2025), https://www.w3.org/TR/2025/CRD-vc-di-bbs-20250403/, uRL: https://www.w3.org/TR/2025/CRD-vc-di-bbs-20250403/
5. Brands, S.: A technical overview of digital credentials (2002), https://api.semanticscholar.org/CorpusID:18284690
6. Braun, C.H., Horne, R., Käfer, T., Mauw, S.: Ssi, from specifications to protocol? formally verify security! In: Chua, T., Ngo, C., Kumar, R., Lauw, H.W., Lee, R.K. (eds.) Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, May 13-17, 2024. pp. 1620–1631. ACM (2024). https://doi.org/10.1145/3589334.3645426, https://doi.org/10.1145/3589334.3645426
7. Braun, C.H., Käfer, T.: RDF-Based Semantics for Selective Disclosure and Zero-Knowledge Proofs on Verifiable Credentials. In: Curry, E., Acosta, M., Poveda-Villalón, M., van Erp, M., Ojo, A.K., Hose, K., Shimizu, C., Lisena, P. (eds.) The Semantic Web - 22nd European Semantic Web Conference, ESWC 2025, Portoroz, Slovenia, June 1-5, 2025, Proceedings, Part I. Lecture Notes in Computer Science, vol. 15718, pp. 383–402. Springer (2025). https://doi.org/10.1007/978-3-031-94575-5_21, https://doi.org/10.1007/978-3-031-94575-5_21
8. Bureau, J.: How Data-Driven Transparency Can Help Safeguard The Supply Chain (sep 2025), https://www.forbes.com/councils/forbesbusinesscouncil/2025/09/17/how-data-driven-transparency-can-help-safeguard-the-supply-chain/
9. Camenisch, J., Drijvers, M., Lehmann, A.: Anonymous attestation using the strong diffie hellman assumption revisited. IACR Cryptol. ePrint Arch. p. 663 (2016), http://eprint.iacr.org/2016/663
10. Camenisch, J., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Revised Papers of the 3rd SCN. LNCS, vol. 2576, pp. 268–289. Springer (2002)
11. Camenisch, J., Stadler, M.: Proof systems for general statements about discrete logarithms. In: Fumy, W. (ed.) Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceedings. Lecture Notes in Computer Science, vol. 1233, pp. 311–323. Springer (1997). https://doi.org/10.1007/3-540-62975-0_25

12. Campanelli, M., Fiore, D., Querol, A.: LegoSNARK: Modular design and composition of succinct zero-knowledge proofs. IACR Cryptol. ePrint Arch. p. 142 (2019)
13. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. IACR Cryptol. ePrint Arch. p. 140 (2002), http://eprint.iacr.org/2002/140
14. Carroll, J.J., Bizer, C., Hayes, P.J., Stickler, P.: Named graphs, provenance and trust. In: Ellis, A., Hagino, T. (eds.) Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14, 2005. pp. 613–622. ACM (2005). https://doi.org/10.1145/1060745.1060835, https://doi.org/10.1145/1060745.1060835
15. Cyganiak, R., Wood, D., Lanthaler, M.: Rdf 1.1 concepts and abstract syntax. W3C Recommendation, W3C (2014), https://www.w3.org/TR/rdf11-concepts/
16. Eagen, L., Kanjalkar, S., Ruffing, T., Nick, J.: Bulletproofs++: Next generation confidential transactions via reciprocal set membership arguments. IACR Cryptol. ePrint Arch. p. 510 (2022), https://eprint.iacr.org/2022/510
17. European Commission: EU Digital Identity Wallet (2024), https://ec.europa.eu/digital-building-blocks/sites/display/EUDIGITALIDENTITYWALLET/, accessed: 2024-09-17
18. European Commission: Regulation (eu) 2024/1183 of the european parliament and of the council amending regulation (eu) no 910/2014 as regards establishing the european digital identity framework (2024), https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=OJ:L_202401183, accessed: 2024-09-17
19. Flouris, G., Fundulaki, I., Pediaditis, P., Theoharis, Y., Christophides, V.: Coloring rdf triples to capture provenance. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) The Semantic Web - ISWC 2009. pp. 196–212. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
20. Geerts, F., Unger, T., Karvounarakis, G., Fundulaki, I., Christophides, V.: Algebraic structures for capturing the provenance of sparql queries. J. ACM **63**(1) (Feb 2016). https://doi.org/10.1145/2810037, https://doi.org/10.1145/2810037
21. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems. In: Proc. of the 17th ACM STOC. pp. 291–304. ACM (1985)
22. Graux, H.: The symbiosis between data protection and open data: A primer on how to reconcile eu data protection law with open data policies. Tech. Rep. OA-01-24-039-EN-Q, Publications Office of the European Union, Luxembourg (jul 2024). https://doi.org/10.2830/1874306, prepared for data.europa.eu, an initiative of the European Commission
23. Green, T.J., Karvounarakis, G., Tannen, V.: Provenance semirings. In: Proceedings of the Twenty-Sixth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. p. 31–40. PODS '07, Association for Computing Machinery, New York, NY, USA (2007). https://doi.org/10.1145/1265530.1265535, https://doi.org/10.1145/1265530.1265535
24. Harris, S., Seaborne, A.: Sparql 1.1 query language. W3C Recommendation, W3C (2013), https://www.w3.org/TR/sparql11-query/
25. Hernández, D., Galárraga, L., Hose, K.: Computing how-provenance for sparql queries via query rewriting. Proc. VLDB Endow. **14**(13), 3389–3401 (Sep 2021). https://doi.org/10.14778/3484224.3484235, https://doi.org/10.14778/3484224.3484235
26. Kierkegaard, P.: E-prescription across europe. Health Technology **3**, 205–219 (2013). https://doi.org/10.1007/s12553-012-0037-0

27. Kilian, J.: Uses of randomness in algorithms and protocols. MIT Press (1990)
28. Knublauch, H., Sommer, A.: SHACL 1.2 SPARQL extensions. Working draft, World Wide Web Consortium (W3C) (Nov 2025), https://www.w3.org/TR/2025/WD-shacl12-sparql-20251125/, latest version available at https://www.w3.org/TR/shacl12-sparql/
29. Kuhn, T., Chichester, C., Krauthammer, M., Queralt-Rosinach, N., Verborgh, R., Giannakopoulos, G., Ngomo, A.N., Viglianti, R., Dumontier, M.: Decentralized provenance-aware publishing with nanopublications. PeerJ Comput. Sci. **2**, e78 (2016)
30. Li, X., Weng, C., Xu, Y., Wang, X., Rogers, J.: ZKSQL: verifiable and efficient query evaluation with zero-knowledge proofs. Proc. VLDB Endow. **16**(8), 1804–1816 (2023). https://doi.org/10.14778/3594512.3594513, https://www.vldb.org/pvldb/vol16/p1804-li.pdf
31. Lipmaa, H.: On black-box knowledge-sound commit-and-prove snarks. In: Guo, J., Steinfeld, R. (eds.) Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4-8, 2023, Proceedings, Part II. Lecture Notes in Computer Science, vol. 14439, pp. 41–76. Springer (2023). https://doi.org/10.1007/978-981-99-8724-5_2
32. Longley, D., Sporny, M.: RDF dataset canonicalization. Final community group report, W3C (2022), https://www.w3.org/community/reports/credentials/CG-FINAL-rdf-dataset-canonicalization-20221009/
33. Morel, S.: Computational integrity for outsourced execution of SPARQL queries. Master's dissertation, Ghent University (2019)
34. Mulder, G.D., Dedecker, R., Meester, B.D., Colpaert, P.: Towards queryable verifiable credentials. In: ISWC 2025 Companion Volume. Nara, Japan (Nov 2025)
35. Pointcheval, D., Sanders, O.: Short randomizable signatures. Cryptology ePrint Archive, Paper 2015/525 (2015), https://eprint.iacr.org/2015/525
36. Shpilka, A., Yehudayoff, A.: Arithmetic circuits: A survey of recent results and open questions. Found. Trends Theor. Comput. Sci. **5**(3-4), 207–388 (2010). https://doi.org/10.1561/0400000039, https://doi.org/10.1561/0400000039
37. Sion, L., Landuyt, D.V., Wuyts, K., Joosen, W.: Robust and reusable LINDDUN privacy threat knowledge. Comput. Secur. **154**, 104419 (2025). https://doi.org/10.1016/J.COSE.2025.104419, https://doi.org/10.1016/j.cose.2025.104419
38. Sporny, M., Noble, G., Longley, D., Burnett, D.C., Zundel, B., Hartog, K.D.: Verifiable credentials data model v1.1. W3C recommendation, W3C (2022), https://www.w3.org/TR/vc-data-model/
39. Terbu, O., Lodderstedt, T., Yasuda, K., Fett, D., Heenan, J.: OpenID for Verifiable Presentations 1.0. Technical specification, OpenID Digital Credentials Protocols Workgroup (July 2025), final Specification
40. Vitto, G., Biryukov, A.: Dynamic universal accumulator with batch update over bilinear groups. IACR Cryptol. ePrint Arch. p. 777 (2020), https://eprint.iacr.org/2020/777
41. Wright, J.: Towards provable provenance and privacy-preserving queries in decentralised data architectures. In: ISWC 2025 Companion Volume. Nara, Japan (Nov 2025)
42. Wu, H., Wei, C., Wang, Y., Lin, L., Leng, Y., He, S., Zhao, M., Wu, H., Yan, Y., Zhou, A.: Zero-knowledge verifiable graph query evaluation via expansion-centric operator decomposition. CoRR **abs/2507.00427** (2025).

https://doi.org/10.48550/ARXIV.2507.00427, https://doi.org/10.48550/arXiv.2507.00427

43. Yamamoto, D., Suga, Y., Sako, K.: Formalising linked-data based verifiable credentials for selective disclosure. In: IEEE European Symposium on Security and Privacy, EuroS&P 2022 - Workshops, Genoa, Italy, June 6-10, 2022. pp. 52–65. IEEE (2022). https://doi.org/10.1109/EUROSPW55150.2022.00013, https://doi.org/10.1109/EuroSPW55150.2022.00013
44. Zhou, E., Guo, S., Hong, Z., Jensen, C.S., Xiao, Y., Zhang, D., Liang, J., Pei, Q.: Veridkg: A verifiable SPARQL query engine for decentralized knowledge graphs. Proc. VLDB Endow. **17**(4), 912–925 (2023). https://doi.org/10.14778/3636218.3636242, https://www.vldb.org/pvldb/vol17/p912-zhou.pdf