A Single-Prover Zero-Knowledge Architecture for Verifiable SPARQL over Committed RDF Graphs

Jesse Wright · the sparq project

A systems-and-design contribution under an open audit gate. It describes the architecture of a single-prover zero-knowledge query-answering stack and reports deterministic artifact facts about it (circuit-family gate counts). It is not *a security, soundness, privacy, or attestation proof: the verifier is* internally re-audited but not externally audited *(the accredited-cryptographer review, gate* `sq-qhy4`*, is* ***open****), the hidden-holder tiers are explicitly* not yet sound*, and the dual-leaf value lane carries an accepted invariant downgrade. No property below may be read as a settled guarantee. "Soundness" and "binding" are used as the technical* nouns *of the goals the design reaches for, never as achieved claims. Wall-clock proving cost is* non-canonical *on our development host and is presented only as a model driven by the gate counts (§7).*

Abstract

Federated SPARQL assumes every endpoint discloses its solution mappings in cleartext. A single data holder who wants to answer a query over credentials it holds — proving the answer follows from *attested* data without revealing the data — needs a different machine: a zero-knowledge proof that a SPARQL result is a genuine evaluation over committed graphs. We describe the architecture of one such single-prover stack. Its shape is deliberate: a per-graph algebraic commitment over the RDF-canonical form; a *fixed, named family* of 14 circuit kinds (35 compiled members across their size lattices), each proving one operator instance of a small monotone SPARQL fragment; a JSON *manifest* that composes sub-proofs through binding edges; and a verifier that *re-derives* the circuit identity and the claimed statement from the query text and the relying party's trust anchors, trusting nothing the prover declares. We give the fragment and its algebraic semantics, the commitment and attestation layer, the manifest composition model, and the verifier's 12-obligation fail-closed pipeline organised around 4 cross-cutting audit gates. We report the family's cost as deterministic `bb` gate counts — the scan lattice spans 5991–34821 gates and the composable filter lanes are gate-identical at 17416 — and derive a proving cost model from them, because we hold our own wall-clock numbers to be non-canonical. On security we are equally precise: the design has survived an internal adversarial audit that found and remediated 12 issues, each now pinned closed by a standing forge-negative regression test, but it has *no* external audit, so we claim no proven property and treat the internal audit as necessary and explicitly not sufficient.

1. Introduction

A verifiable credential lets a holder prove a fact an issuer signed. But real questions are rarely a single signed fact — they are *queries*: "does the union of my credentials contain a person over 18 whose employer is on this list?" Answering such a question while disclosing only the answer, and proving the answer is a faithful SPARQL evaluation over *attested* data, is the problem this architecture addresses. It is the single-holder, single-prover case: one party holds the graphs, commits to them, and produces one proof a relying party checks.

The problem is hard for three reasons that a naive design misses. First, *the prover is the adversary.* Everything in a proof request is prover-controlled — the query text, the declared circuit, the public inputs, the verification key, even the freshness nonce if the design lets it. A verifier that trusts any prover-declared quantity has no soundness to speak of. Second, *composition is where forgeries live.* A result over "age ≥ 18 AND employer ∈ S" is not one proof but several — a scan, a filter, a join — and the dangerous attacks are not against a single circuit but against the *seams*: proving `17 ≥ 17` for a `≥ 18` query, pointing a filter edge at the wrong column, or replaying an honest proof of a true-but-different statement. Third, *the RDF data model adds its own hazards* — blank-node identity is graph-scoped, so

a join that correlates blank nodes across two committed graphs is semantically meaningless and must be excluded, not silently admitted.

Why has this not been solved off-the-shelf? Verifiable-database systems (IntegriDB [1], vSQL [2], ZKSQL [3]) prove SQL over a *known* database to a client that already trusts the schema; they do not hide the graph from the verifier, and they do not bind results to *issuer attestations*. Anonymous-credential systems [4], [5] prove signed attributes but not *query evaluation* over a set of credentials. The closest semantic-web work proves selective-disclosure soundness of SPARQL results [6]; this paper is a companion *systems* description of an independently-built engine-integrated stack and its cost and audit posture, framed under an open external-audit gate.

This paper contributes:

- **The provable fragment and its semantics** (§2.) — a monotone, open-world-conforming SPARQL subset (BGP scans, datatype-bucketed value `FILTER`, a hidden-credential equality `JOIN`, membership-indifferent modifiers), pinned to the Pérez–Arenas–Gutiérrez algebra [7] with a stated *result-membership* correctness target and an explicit cross-graph blank-node exclusion.
- **The commitment and attestation layer** (§3.) — per-graph Poseidon2 [8] commitments over the RDFC-1.0 [9] canonical form, bound to issuer keys by a Schnorr-over-Baby-Jubjub [10], [11] attestation, so a prover cannot be the issuer of its own facts.
- **The fixed named circuit family and the manifest composition model** (§4.) — 14 circuit kinds, 35 compiled members, composed by a manifest whose binding edges chain sub-proofs; and **the verifier re-derivation discipline** (§5.) — the 12 fail-closed obligations and 4 audit gates that reconstruct the statement from trust anchors rather than trust the manifest.
- **A deterministic cost characterisation and a proving-cost model** (§6.) — `bb` UltraHonk gate counts from the regression-gated snapshot, and a proving-cost model over them, with an explicit statement that our own wall-clock numbers are non-canonical.
- **An honest security analysis under an open gate** (§7.) — the threat model, the 12 findings of an internal adversarial audit and their remediations pinned by 12 standing forge-negative regression tests, and a precise statement of what the *absence* of an external audit (`sq-qhy4`) means for every claim.

2. The provable SPARQL fragment

The fragment is deliberately small, and the smallness is principled rather than incidental. The target correctness property is *result membership*: a manifest disclosing a solution mapping μ (or asserting one exists) for a pattern P over committed graphs $G_1 \ldots G_n$ is correct iff $\mu \in \text{eval}(P)$. The admission rule is: include exactly the constructs for which membership is *monotone* — a witness that survives the world gaining more data. That is the operational content of "conforms to the open-world assumption", and it is exactly the deployment model, in which a holder presents a *subset* of its credentials.

Grammar and algebra. Following the SPARQL 1.1 algebra of Pérez, Arenas and Gutiérrez [7] over the RDF 1.1 graph model [12], a fragment pattern is

$$P ::= \text{BGP} \mid \text{Filter}(C, P) \mid \text{Join}(P_1, P_2)$$

where a BGP is evaluated against *exactly one* committed graph, `C` is a datatype-bucketed value constraint, and `Join` is an equality join whose two sub-patterns range over *distinct* committed graphs and share a variable. Evaluation is the standard set semantics, with SPARQL expression errors in a `Filter` treated as *not satisfied* [13].

Construct	Disposition	Why
`SELECT` / `ASK`	In	Membership / non-emptiness of eval(P); monotone.
BGP scan	In	Row soundness + per-scan completeness proved in-circuit.
Value `FILTER`	In (4+ datatype lanes)	Monotone under error-as-unsatisfied semantics.
Equality `JOIN`	In	Hidden-credential join key; cross-graph blank-node join excluded.
`DISTINCT`/`REDUCED`/`LIMIT`/`OFFSET`/projection	In	Membership-indifferent modifiers.
`OPTIONAL` / `MINUS` / `NOT EXISTS`	Out	Non-monotone: a closed-world "no extension exists" claim.
Aggregation / `GROUP BY`	Out	An aggregate is a whole-pattern completeness claim — closed-world.
`GRAPH`	Out	Naming graphs discloses the attribution the model hides.
`SERVICE`	Out	Federation is out of scope by construction.
`ORDER BY`	Out	Membership-indifferent, but a reader infers an unproved top-k claim.

Table 1: The provable fragment. Constructs are admitted on *semantics* (monotone result membership), not on circuit cost; the closed-world / completeness-dependent operators are excluded because additional undisclosed data could falsify a "proved" answer. This is the maximal-monotone position; property paths and most filter *expressions* are a designed but not-yet-implemented extension (bead `sq-3kd2g`), out of scope for this paper's architecture.

Cross-graph blank nodes. Blank-node identity is scoped to a single graph [12], and per-graph canonicalisation cannot align blank-node labels *across* graphs. A `Join` solution binding a shared variable to a blank node in more than one committed graph is therefore *excluded* from eval; the architecture enforces this exclusion (the "Q6" guard) rather than admitting a correlation the data model does not support.

3. Commitment and attestation

Per-graph commitment. Each source graph is canonicalised with RDF Dataset Canonicalization (RDFC-1.0) [9] — so the commitment is independent of blank-node labelling and triple order — and committed with the Poseidon2 permutation [8] over the BN254 scalar field, one commitment per graph. Poseidon2 is chosen for in-circuit efficiency (it is arithmetization-friendly, unlike a byte-oriented hash); the commitment is a standard binding commitment in the sense of Pedersen [14], a design goal the architecture reaches for and does not claim as an audited property.

Issuer attestation. A commitment alone lets a prover commit to any graph it invents — it would be the issuer of its own facts. The architecture therefore binds each commitment to an issuer key by a Schnorr signature [10] over the Baby-Jubjub curve [11] with a Poseidon2-derived challenge. The verifier accepts a key only if the relying party placed it in an *external* trusted key set K (§5.) — never merely because the manifest lists it. This attestation layer is the design's answer to the single most severe class of the internal audit (§7.); it *transfers* trust from issuer to result, it does not create trust in the issuer's real-world honesty, which is out of cryptographic scope.

Post-quantum posture (a settled negative). The signature and the commitment binding rest on discrete-log and hash assumptions; the Schnorr/Baby-Jubjub attestation falls to a Shor-capable adversary, so this stack offers no post-quantum guarantee and we state so plainly rather than imply resilience.

4. The circuit family and the manifest

Every sub-proof is generated against exactly one circuit of a *fixed, named* family — this is the load-bearing architectural choice. A fixed family means the verifier can re-derive *which* circuit a statement demands and recompute *that* circuit's verification key, instead of trusting a prover-supplied key over a prover-chosen circuit. The alternative — synthesising a bespoke circuit per query — would require a circuit-identity-to-query binding the verifier could check, a much larger trust-model surface, and is deliberately not taken here.

Circuit kind	Statement proved (descriptive gloss)
`Scan`	A BGP scan matches against a committed graph (row soundness + per-scan completeness).
`FilterInt` / `FilterF64` / `FilterSignedInt` / `FilterDecimal`	A datatype-bucketed value FILTER holds, operand bound to the committed literal.
`FilterValueDl*`	An opt-in dual-leaf value-lane FILTER (accepted invariant downgrade; off by default).
`JoinEq`	Two hidden credentials agree on an equality join key.
`RevokeUnset`	A revocation bit is unset in a committed status snapshot.
`HiddenIssuer`	The issuer of a hidden credential lies in an attested key set.
`HolderPok` / `HolderSet`	Hidden-holder binding tiers — explicitly *not yet sound*; opt-in only.

Table 2: The 14 circuit kinds of the fixed family (35 compiled members across their size lattices). Each kind realises one operator instance of the fragment (§2.) or one auxiliary statement. The circuit identifier a manifest declares is re-derived by the verifier and never trusted on its own (§5.). The hidden-holder kinds are labelled not-yet-sound in the implementation and are gated off by default.

The manifest. A proof is a JSON *manifest* carrying the key set, the sub-proofs (each a length-prefixed proof, public-input segment, and verification key), the attribution set relating result rows to source graphs, and the binding material the verifier's obligations consume. The verifier nonce is committed as public-input field 0 of *every* sub-proof, so a manifest is bound to a single request. Composition across

operators is expressed by *binding edges*: an edge asserts that, e.g., the scanned column a filter constrains equals the filter's operand, chaining sub-proofs into one statement. The binding-edge mechanism is exactly where composition attacks live, and it was itself the subject of an internal adversarial review — the analysis in §7. is organised around it.

5. Verification: re-derive, never trust

The verifier's discipline is a single principle: *reconstruct the claimed statement from the query text and the relying party's trust anchors, and check the cryptography against the reconstruction* — trusting no prover-declared quantity. It runs fail-closed: the first failed check rejects the whole manifest, with no partial results and no downgrade to a warning.

The pipeline enforces 12 binding obligations, structured around 4 cross-cutting *audit gates* whose individual failure would each void the intended soundness on its own:

Audit gate	What the verifier does instead of trusting the manifest
1 — public-input reconstruction	Independently reconstructs every sub-proof's expected public-input bytes — with the verifier nonce at field 0 — from the declared statement and compares byte-for-byte; any difference rejects. Without this, an honest proof of a *different* true statement is replayable as a forgery.
2 — canonical verification key	Recomputes each sub-proof's verification key from the canonical circuit named by the *re-derived* identifier; a manifest-supplied key is never trusted. Without this, a prover-chosen key over an unconstrained circuit defeats the whole gate.
3 — issuer signature and key set	Requires every issuer key used to be a member of the *external* key set K, and the issuer signature over the graph commitment to verify. Without this, the prover is the issuer of its own facts.
4 — nonce single-use and binding	Mints a fresh single-use nonce, records it as burnt *before* the cryptographic checks, and rejects any manifest that binds a different nonce. Without this, an accepting manifest is replayable forever.

Table 3: The 4 audit gates. Backend proof verification is layered *around* them: verifying a proof is meaningless unless gates 1 and 2 pin *what* is being verified and *against which key*. The gates are internally re-audited, *not* externally audited (`sq-qhy4`, open); this table describes the mechanism, not a proven guarantee.

Beyond the gates, the obligation set re-derives the circuit identifier from the statement, re-checks the cross-graph blank-node exclusion of §2., enforces the attribution superset rule binding result rows to source graphs, binds filter operators/bounds and join keys to the query text, and re-checks any declared RDFS/OWL derivation steps against *disclosed* bases (only simple entailment is proved in zero knowledge; an in-circuit closure proof is deferred). Each failure maps to an explicit variant of a closed error taxonomy.

6. Cost characterisation and a proving-cost model

We characterise the family's cost by its *deterministic artifact facts*: `bb gates -s ultra_honk` circuit sizes, the ground-truth constraint count under the pinned toolchain, taken from the regression-gated snapshot (so the numbers cannot silently drift). These are machine-*independent* integer facts of the compiled circuits — not timings — and are the only cost numbers this paper treats as canonical.

Family member (representative)	UltraHonk gates
`RevokeUnset` (revocation, depth 10)	899
`FilterValueDl` (opt-in dual-leaf integer lane)	3033
`Scan` — smallest (k=1, n=16, r=4)	5991
`JoinEq` — smallest (n_a=16, n_b=16)	7025
`HolderPok` (hidden-holder, not-yet-sound)	10334
`HiddenIssuer` (in-circuit Schnorr + key-set membership)	24452
`JoinEq` — largest (n_a=64, n_b=64)	18681
Composable filter lane (`filter_int`/`filter_f64`/… any digit count)	17416
`Scan` — largest (k=2, n=64, r=8)	34821

Table 4: Deterministic circuit sizes across the family, from the regression-gated gate-count snapshot (`bb gates -s ultra_honk`, toolchain pinned to `bb 5.0.0-nightly.20260324` / `nargo 1.0.0-beta.21`). Two facts drive the design's cost story: the string-canonical filter lanes are *gate-identical* at 17416 regardless of digit count (the blake3 binding of the canonical literal token dominates and fits one hash block), and the scan members scale with the (k·n) commitment-recompute sweep. The opt-in dual-leaf lane (3033) shows the cost the blake3 binding buys — it is cheaper but carries an accepted invariant downgrade and is off by default.

evidence: `crates/sparq-zk-compose/tests/gate_count_snapshot.json::members['scan_k2_n64_r8']` (`bb gates -s ultra_honk; regression-gated by tests/gate_count.rs, 3% tolerance`) (environment: canonical)

A proving-cost model, and why we publish a model rather than a headline time. UltraHonk proving cost is, to first order, linear in the circuit's gate count for a fixed backend and thread count; proof size and verification cost are *constant* across the family (the succinctness property of the scheme). So the gate-count table above *is* the cost profile up to a single machine-dependent constant: prove-time $\approx \kappa \cdot$ gates for a per-host κ, with verification and proof size flat. We do *not* publish a headline wall-clock number because our development measurements are taken on an AWS EC2 host whose timings are non-canonical under the project's empirical-honesty mandate — a speed claim would require the canonical runner, which is not yet available for this family. Relative cost within the family, however, is a property of the gate counts and *is* canonical: a largest-scan proof carries about 5.8× the constraints of the smallest scan, and about 2× a composable filter lane — ratios a reader can multiply by any host's measured κ. This is the honest form of a cost result when the absolute timer is not yet canonical: publish the constraint counts and the linear model, and name the missing constant.

7. Security analysis under an open audit gate

We are precise about status because the framing is the contribution. The architecture is *research-grade* and has *no* external accredited-cryptographer audit; the external audit is an open gate (`sq-qhy4`). Nothing below is a proven guarantee.

Threat model. The prover is fully adversarial: it controls the manifest, the query text, the declared circuit and public inputs, the verification key, and any replayed material, and its goal is to make the verifier accept a false SPARQL statement, reuse a proof, or smuggle in an untrusted issuer. The verifier is honest-but-curious for privacy and is trusted by its relying party to run the whole obligation set. Issuer content veracity, side channels, and transport are out of scope.

The internal adversarial audit. An internal adversarial audit of the verifier found and confirmed 12 issues on a v1 verifier — that v1 was documented *not* sound. The confirmed classes were exactly the composition-seam attacks the architecture must defend: public inputs never reconstructed from the declared statement, a prover-supplied verification key trusted as-is, commitments accepted without

an issuer signature, manifests infinitely replayable for want of a nonce binding, and filter operator/bound/slot never bound to the query's FILTER. Each has a stated remediation — the four audit gates of §5. are, in large part, that remediation — and each finding now carries a standing *forge-negative* regression test:

Committed structural fact	Count
Confirmed findings in the internal single-prover verifier audit	12
Findings pinned closed by a 1:1 forge-and-verify regression test	12
Fail-closed binding obligations enforced per manifest	12
Cross-cutting audit gates	4

Table 5: The audit posture as *deterministic structural facts* (source/document scans over committed code, not measurements). The 12 forge-negative tests each construct a specific historical forgery and assert the verifier rejects it with the mapped error, so a future refactor cannot silently re-open a remediated finding. This is *necessary and explicitly not sufficient* evidence: it pins known attacks closed; it does not discover unknown ones, which is the job of the open external audit.

evidence: `crates/sparq-zk-compose/tests/audit_forge_map.rs (bead sq-1gir): one named forge-and-verify regression test per historical audit finding #1-#12, each constructing that finding's specific forgery and asserting the mapped verifier reject path` (environment: canonical)

What the absence of an external audit means. A passing verification must *not* be read as a settled guarantee against an adversarial prover. The internal audit and the forge-negative suite raise assurance and are honestly reported as such; they are not a proof of soundness, and no accredited cryptographer has reviewed the estate. Positive security properties are therefore at most *claimed*, with audit status "external sign-off pending". The hidden-holder tiers (`HolderPok`, `HolderSet`) are explicitly *not yet sound* and are off by default; the dual-leaf value lane carries an accepted, documented invariant downgrade and is opt-in; only simple entailment is in zero knowledge. We report each of these with equal precision to the design's strengths, because an honest architecture paper under an open gate is honest only if it does.

Relation to the collaborative path. This paper is strictly single-prover. The *multi-prover* (collaborative) extension — several holders jointly proving over their private graphs — is a separate, *unbuilt* path with its own open gate and its own negative-result analysis against the collaborative-zk-SNARK failure modes of Garg et al. [15]; it is out of scope here and claims nothing.

8. Related work

Verifiable databases. IntegriDB [1], vSQL [2], and ZKSQL [3] prove SQL query results over an outsourced database. They target a different trust shape: the querier trusts the schema and wants integrity of an untrusted *server's* computation; they do not hide the data from the verifier, do not operate over RDF, and do not bind results to issuer attestations. Our architecture hides the committed graphs and roots results in signed issuer keys.

Anonymous credentials. CL signatures [4], BBS [16], and zk-creds [5] prove possession of signed attributes with selective disclosure. They prove facts about *one credential's fields*, not *query evaluation* over a set of credentials with joins and filters — which is the structure this fragment adds.

Signed RDF and semantic-web ZK. Signing RDF graphs is classical [17]; the canonicalisation this architecture relies on is the modern RDFC-1.0 [9]. The most direct neighbour proves selective-disclosure soundness of SPARQL results over RDF datasets with zero-knowledge proofs [6]; this paper is a companion systems description of an engine-integrated single-prover stack — its fixed-circuit-family architecture, its verifier re-derivation discipline, its deterministic cost profile, and its audit posture

under an open gate. We claim no cryptographic novelty over the primitives we compose; the contribution is the system and its honest characterisation.

9. Conclusion

A single holder answering a SPARQL query in zero knowledge over attested credentials needs a machine whose every part is built for an adversarial prover: a per-graph commitment bound to an issuer signature, a *fixed named family* of 14 circuit kinds (35 compiled members) so the verifier can recompute keys rather than trust them, a manifest that composes sub-proofs through checkable binding edges, and a 12-obligation, 4-audit-gate verifier that re-derives the claimed statement from the query and the relying party's trust anchors. We reported the family's cost as deterministic gate counts (5991–34821 for the scan lattice, 17416 for the composable filter lanes) and a linear proving-cost model over them, and we named the one missing piece — a canonical wall-clock constant — rather than quote a non-canonical time. On security we reported an internal audit that found and remediated 12 issues, each pinned closed by a standing forge-negative test, and we stated plainly that with *no* external audit (`sq-qhy4`) the design claims no proven property. The architecture is what it is: a carefully-composed, internally re-audited, not-yet-externally-audited system, described so that both its design and its open gate are legible.

References

[1] Y. Zhang, J. Katz, and C. Papamanthou, "IntegriDB: Verifiable SQL for Outsourced Databases," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2015.

[2] Y. Zhang, D. Genkin, J. Katz, D. Papadopoulos, and C. Papamanthou, "vSQL: Verifying Arbitrary SQL Queries over Dynamic Outsourced Databases," in *IEEE Symposium on Security and Privacy (S&P)*, 2017.

[3] X. Li, C. Weng, Y. Xu, X. Wang, and J. Rogers, "ZKSQL: Verifiable and Efficient Query Evaluation with Zero-Knowledge Proofs," *Proceedings of the VLDB Endowment*, vol. 16, 2023.

[4] J. Camenisch and A. Lysyanskaya, "An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation," in *Advances in Cryptology — EUROCRYPT 2001*, 2001.

[5] M. Rosenberg, J. White, C. Garman, and I. Miers, "zk-creds: Flexible Anonymous Credentials from zkSNARKs and Existing Identity Infrastructure," in *IEEE Symposium on Security and Privacy (S&P)*, 2023.

[6] C. H.-J. Braun, J. Wright, and T. Käfer, "Proving Soundness of SPARQL Query Results Using Selective Disclosure of RDF Datasets and Zero-Knowledge Proofs," in *The Semantic Web (ESWC 2026)*, 2026.

[7] J. Pérez, M. Arenas, and C. Gutiérrez, "Semantics and Complexity of SPARQL," *ACM Transactions on Database Systems*, vol. 34, no. 3, 2009.

[8] L. Grassi, D. Khovratovich, and M. Schofnegger, "Poseidon2: A Faster Version of the Poseidon Hash Function," in *Progress in Cryptology — AFRICACRYPT 2023*, 2023.

[9] D. Longley, G. Kellogg, and D. Yamamoto, "RDF Dataset Canonicalization." W3C, 2024. [Online]. Available: https://www.w3.org/TR/rdf-canon/

[10] C.-P. Schnorr, "Efficient Signature Generation by Smart Cards," *Journal of Cryptology*, vol. 4, no. 3, 1991.

[11] M. Bellés-Muñoz and J. Baylina, "EIP-2494: Baby Jubjub Elliptic Curve." 2020. [Online]. Available: https://eips.ethereum.org/EIPS/eip-2494

[12] R. Cyganiak, D. Wood, and M. Lanthaler, "RDF 1.1 Concepts and Abstract Syntax." W3C, 2014. [Online]. Available: https://www.w3.org/TR/rdf11-concepts/

[13] S. Harris and A. Seaborne, "SPARQL 1.1 Query Language." W3C, 2013. [Online]. Available: https://www.w3.org/TR/sparql11-query/

[14] T. P. Pedersen, "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing," in *Advances in Cryptology — CRYPTO '91*, 1991.

[15] S. Garg, A. Goel, A. Jain, B. Roberts, and S. Sekar, "Malicious Security in Collaborative zk-SNARKs: More than Meets the Eye," 2025. [Online]. Available: https://eprint.iacr.org/2025/1026

[16] T. Looker, V. Kalos, A. Whitehead, and M. Lodder, "The BBS Signature Scheme." IRTF CFRG Internet-Draft, 2025. [Online]. Available: https://datatracker.ietf.org/doc/draft-irtf-cfrg-bbs-signatures/

[17] J. J. Carroll, "Signing RDF Graphs," in *The Semantic Web — ISWC 2003*, 2003.

---

sparq project. This paper is a systems-and-design contribution under the OPEN external-audit gate `sq-qhy4`; it asserts *no* proven security, soundness, privacy, or attestation property. Evidence traces to the fixed circuit family `zk/compose/`, the regression-gated gate-count snapshot `crates/sparq-zk-compose/tests/gate_count_snapshot.json`, the verifier `crates/sparq-zk-compose/src/verifier.rs`, the internal audit `research/zk-soundness-audit.md`, and the forge-negative regression map `crates/sparq-zk-compose/tests/audit_forge_map.rs`. Numbers are injected at build time from the paper-bound evidence file; see the provenance stamp on the published page.