PSolid: A Web-Native Framework for Privacy-Preserving and Sovereign Data Verification

Francisco Javier Becerra Sanchez
University of Luxembourg
Luxembourg, Luxembourg
francisco.becerra@uni.lu

Antonio Ken Iannillo
University of Luxembourg
Luxembourg, Luxembourg
antonioken.iannillo@uni.lu

Enriqueta Patricia Becerra Sanchez
University of Luxembourg
Luxembourg, Luxembourg
patricia.becerra@uni.lu

Radu State
University of Luxembourg
Luxembourg, Luxembourg
radu.state@uni.lu

Abstract

The modern Web is rapidly evolving toward a decentralized, user-centric ecosystem where data sovereignty and privacy are essential foundations. Solid (Social Linked Data), a W3C initiative, embodies this paradigm by allowing users to manage personal data through interoperable, Web-based storage pods. However, once access is granted, Solid provides no protection against data misuse, leaving users vulnerable to unauthorized data replication and privacy breaches.

This paper introduces PSolid, an extension of the Solid framework designed to enable privacy-preserving data verification without disclosing sensitive information. PSolid leverages zero-knowledge proofs (SNARKs) and the Edwards-curve Digital Signature Algorithm (EdDSA) to allow third parties to verify user claims securely, while maintaining compliance with data protection standards such as GDPR and ISO/IEC 27001. We validate the system through a real-world Web scenario in which users must prove attributes (e.g., age, income, or credentials) without exposing underlying data. A comprehensive performance evaluation and threat analysis using STRIDE and LINDDUN demonstrate that PSolid effectively preserves data sovereignty, privacy, and integrity in Web-based identity verification.

By extending the Solid ecosystem with cryptographic guarantees of trust and confidentiality, this work contributes to the engineering of a more secure, privacy-aware, and sovereign Web 3.0 infrastructure

CCS Concepts

• **Information systems → Secure online transactions**; *Electronic data interchange.*

Keywords

Privacy-Preserving Verification, Data Sovereignty, Solid, Web Security, Decentralized Web, Zero-Knowledge Proofs

ACM Reference Format:
Francisco Javier Becerra Sanchez, Antonio Ken Iannillo, Enriqueta Patricia Becerra Sanchez, and Radu State. 2026. PSolid: A Web-Native Framework for Privacy-Preserving and Sovereign Data Verification. In *The 41st ACM/SIGAPP Symposium on Applied Computing (SAC '26), March 23–27, 2026, Thessaloniki, Greece.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3748522.3779964

1 Introduction

The Web is gradually shifting from centralized services toward decentralized, user-centric architectures often branded as Web 3.0. In this setting, *data sovereignty*—the ability of users to control where their data resides, who may access it, and under which conditions—is increasingly important [6]. Solid (Social Linked Data) [4, 3] is a W3C initiative that contributes to this vision by separating data from applications: users store information in personal online data stores (Pods) and grant applications access via Web standards such as RDF and Linked Data Platform.

While Solid aligns with privacy regulations such as GDPR and ISO/IEC 27001, it still follows a "read-or-not" paradigm: once an application is authorized, it may copy, aggregate, or redistribute data without further technical restrictions. This is problematic in domains such as housing, employment, or healthcare, where relying parties often need to *verify* properties of a user's data rather than read the data itself.

We address this gap with **PSolid**, a framework that enables third parties to verify statements about Pod-hosted data without direct access to the underlying values. In our running example, a landlord (Verifier) wants to check that an applicant (Prover) earns at least €1500 and has a contract that runs for 12 months, based on a credential issued and signed by an employer (Issuer). Instead of disclosing the full contract, the Prover generates a zero-knowledge proof that the contract stored in their Pod satisfies the landlord's policy.

Contributions. This work formalizes security and privacy requirements for Issuer, Prover, and Verifier; introduces **PSolid**, which integrates EdDSA credentials with Groth16 zk-SNARKs for privacy-preserving verification; implements a Solid-based prototype using Circom and snarkjs; and provides a qualitative STRIDE/LINDDUN analysis with initial performance observations, identifying key limitations and future research directions.

2 PSolid Framework

2.1 Roles, Requirements, and Use Case

PSolid considers three roles: an **Issuer** (e.g., employer) issues digitally signed credentials to a **Prover** (user), who stores them in a Solid Pod; a **Verifier** (e.g., landlord) defines a policy over credential attributes and checks a proof that the Prover's credentials satisfy it.

We group requirements along these roles. For the Issuer, the framework must ensure credential integrity and origin accountability (credentials are signed and bound to a specific subject) and restrict access to the intended Prover (IR1–IR4). For the Prover, it must allow authenticity verification of received credentials, provide privacy-preserving proof of policy compliance, and ensure that the Prover can inspect the Verifier's policy before generating a proof (PR1–PR4). For the Verifier, it must guarantee integrity of the policy, allow verification of compliance without seeing raw data, and bind proofs to trusted Issuers (VR1–VR3).

In the rental use case, the Issuer signs a credential containing attributes such as WebID, salary, contract type, and end date. The Verifier specifies a policy (e.g., JSON) describing required conditions. The Prover then generates a zero-knowledge proof that the credential in their Pod satisfies the policy, and the Verifier checks the proof without downloading the credential.

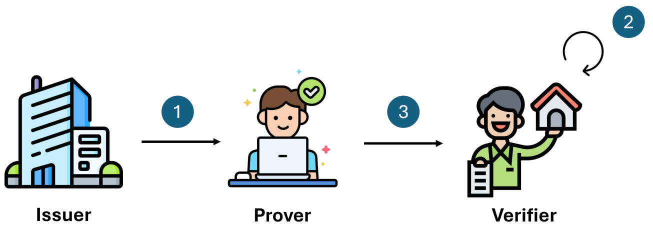

Figure 1: System components and interactions among Issuer, Prover, and Verifier.

2.2 Architecture and Protocol

PSolid builds on three components: first, **Solid Pods** serve as the storage layer for credentials and policies in JSON-LD/RDF form, with access governed by WebID and ACL/OIDC. Second, **EdDSA credentials** are issued by signing a canonical hash of credential attributes, with the resulting credential, signature, and Issuer public key stored in Pods and retrieved over HTTPS. Third, **Groth16 zk-SNARKs** enable the Verifier to encode a policy as a Circom circuit that validates both attribute predicates and the Issuer's signature, while exposing only a policy identifier and the Issuer's public key as public inputs, keeping all attributes private.

The protocol, illustrated in Figure 2, proceeds in four phases:

(1) **Credential issuing.** The Issuer signs the credential and stores it in the Prover's Pod along with the public key. The Prover verifies the signature locally before using the credential.
(2) **Requirement definition.** The Verifier publishes a machine-readable policy (e.g., JSON) in their Pod, then translates it into a Circom circuit and runs the Groth16 trusted setup to obtain a proving key and verification key.

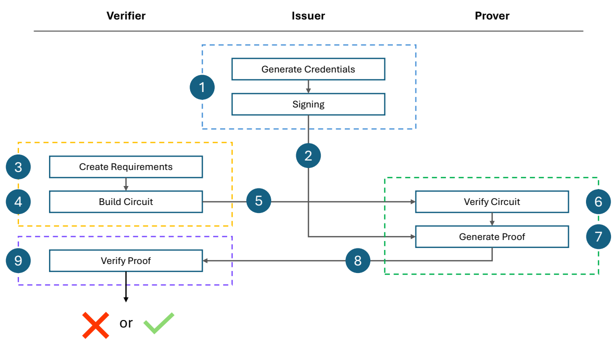

Figure 2: PSolid protocol flow illustrating interactions and data flow among Issuer, Prover, and Verifier. Phases are color-coded: Credential Issuing (blue), Requirement Definition (orange), Proof Generation (green), and Proof Verification (purple).

(3) **Proof generation.** The Prover retrieves the policy and (optionally) recompiles the circuit locally to check that it matches the published specification. Using the Issuer-signed credential as private input, and the policy identifier and Issuer key as public inputs, the Prover computes a witness and generates a zk-SNARK proof.
(4) **Proof verification.** The Verifier checks the proof with the verification key and the expected public inputs. The Verifier learns only whether the policy holds for some Issuer-signed credential; the credential itself never leaves the Prover's Pod.

3 Prototype and Initial Evaluation

We implemented PSolid as a Web application using JavaScript-Node.js and React, interacting with Solid Pods through standard HTTP APIs, and integrating zk-SNARK functionality via Circom 2 and snarkjs (Groth16 over bn-128)[1]. The rental circuit incorporates comparison and equality constraints, a Poseidon hash of attributes, and an EdDSA-on-BabyJub verifier, yielding roughly 10^4 constraints.

Preliminary experiments on a laptop (Intel Core i9-12900H, 32 GB RAM, Windows 10) show that Groth16 trusted setup for the rental circuit takes tens of seconds to a few minutes and constitutes the main cost when introducing a new policy, while proof generation and verification execute in well under a second per proof and remain efficient under moderate concurrency. An earlier observed mismatch between wall-clock and CPU times during setup appears tied to tooling or measurement artifacts rather than Groth16 itself; therefore, we give only approximate timing magnitudes and leave detailed profiling and comparison with alternative schemes (e.g., BBS+ [1]) for future work.

[1]Repository link: https://github.com/franciscobecerra97/PSolid

4 Security, Privacy, and Limitations

We applied STRIDE [5] and LINDDUN [2] qualitatively to the PSolid data flow. Digital signatures, Solid's authentication and authorization, and zk-SNARK-based verification help mitigate spoofing, tampering, information disclosure, and elevation-of-privilege threats. From a privacy perspective, zero-knowledge proofs ensure that only Boolean statements (e.g., "policy satisfied") are leaked, and publishing policies and circuits in Pods improves transparency and informed consent.

Residual risks remain. Stable WebIDs allow correlation of a user's interactions across services, creating linkability. We currently rely on WebID for simplicity; a more privacy-preserving design would integrate Decentralized Identifiers (DIDs) with pairwise-unique keys (e.g., `did:peer`) to reduce cross-service linkability. PSolid also deliberately avoids strong non-repudiation to preserve user autonomy; whether stronger accountability mechanisms should be added is context dependent.

From a practicality standpoint, there are significant limitations:

- **Trusted setup per policy.** Groth16 requires a circuit-specific trusted setup. In our prototype, each new policy entails a new setup, which is a major obstacle for adoption. Future versions of PSolid should investigate proof systems with universal or updatable setups (e.g., PLONK) or transparent, trustless systems (e.g., STARKs).
- **Policy authoring.** Translating human-readable policies into Circom circuits currently requires expert knowledge. A domain-specific language (DSL) and compiler that generate circuits from simple policy statements would be essential for non-expert Verifiers.
- **Circuit cost of EdDSA.** EdDSA verification dominates circuit complexity. Exploring SNARK-friendlier signature schemes or alternative selective-disclosure mechanisms (such as BBS+) could significantly reduce proving cost.
- **Attribute encoding.** Our prototype hashes concatenated attributes before signing; a production system must specify and analyze a canonical encoding to prevent ambiguity and concatenation attacks.

5 Conclusion and Future Work

PSolid demonstrates that it is possible to extend Solid with privacy-preserving, Web-native verification of sovereign user data. By combining Solid Pods, EdDSA-signed credentials, and Groth16 zk-SNARKs, the framework allows Verifiers to check policies over user attributes without accessing the underlying data, and without relying on blockchains or centralized intermediaries.

At the same time, our current design has clear practical limitations, notably the reliance on Groth16 with per-policy trusted setup, the expertise required to author Circom circuits, and open issues in performance measurement and canonical encoding. As future work, we plan to (i) evaluate universal or transparent proof systems, (ii) design a high-level DSL and tooling for policy-to-circuit compilation, (iii) explore alternative signature and disclosure mechanisms, and (iv) integrate pairwise DIDs to reduce linkability. We believe that addressing these challenges will move PSolid from a research prototype toward a practical building block for privacy-preserving, data-sovereign Web applications.

Table 1: Threat Mitigation Overview for PSolid Based on STRIDE and LINDDUN (✓= addressed, != partially addressed, −= not addressed)

Threat Category	Mitigated?	Notes
STRIDE (Security)		
Spoofing (S)	✓	Digital signatures; Solid authentication via WebID and OIDC.
Tampering (T)	✓	Cryptographic integrity checks; secure, access-controlled Pods.
Repudiation (R)	!	Not enforced to preserve data sovereignty and consent.
Information Disclosure (I)	✓	zk-SNARKs prevent credential exposure.
Denial of Service (D)	✓	Decentralized architecture; rate limiting and input validation.
Elevation of Privilege (E)	✓	Credential binding, ACL enforcement, circuit integrity.
LINDDUN (Privacy)		
Linkability (L)	!	Mitigated through decentralization; residual risk via persistent WebIDs.
Identifiability (I)	✓	Zero-knowledge proofs conceal identity and data values.
Non-repudiation (N)	!	Optional to maintain user autonomy and consent.
Detectability (D)	✓	Encrypted, peer-to-peer interactions within Solid Pods.
Disclosure of Information (D)	✓	Only Boolean assertions revealed to the Verifier.
Unawareness (U)	✓	Transparent policies; auditable circuits; informed consent.
Non-compliance (N)	✓	GDPR-aligned via data minimization and transparency.

Acknowledgment

This research was funded in part by the Luxembourg National Research Fund (FNR), grant reference NCER22/IS/16570468/NCER-FT and FinnovationHub funded by the Ministry of Finance, Government of Luxembourg. The author thanks colleagues and reviewers for helpful feedback. Generative AI (OpenAI GPT-5) assisted with clarity and language, while all technical content, methodology, and analysis were solely developed by the author, who remains fully responsible for the ideas and conclusions.

References

[1] C. H.-J. Braun and T. Käfer. 2022. Attribute-based access control on solid pods using privacy-friendly credentials. In *Proceedings of Poster and Demo Track and Workshop Track of the 18th International Conference on Semantic Systems co-located with 18th International Conference on Semantic Systems (SEMANTiCS 2022) Ed.: U. Şimşek* (CEUR Workshop Proceedings). 18th International Conference on Semantic Systems. SEMANTiCS 2022 (Wien, Österreich, Sept. 13–15, 2022). Vol. 3235. CEUR-WS.org, 5 S.

[2] Mina Deng, Kim Wuyts, Riccardo Scandariato, Bart Preneel, and Wouter Joosen. 2011. A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements. *Requir. Eng.*, 16, (Mar. 2011), 3–32. doi:10.1007/s00766-010-0115-7.

[3] Essam Mansour, Andrei Vlad Sambra, Sandro Hawke, Maged Zereba, Sarven Capadisli, Abdurrahman Ghanem, Ashraf Aboulnaga, and Tim Berners-Lee. 2016. A Demonstration of the Solid Platform for Social Web Applications. *Proceedings of the 25th International Conference on World Wide Web (WWW '16)*, (Jan. 2016), 223–226. doi:10.1145/2872518.2890529.

[4] Andrei Vlad Sambra, Essam Mansour, Sandro Hawke, Maged Zereba, Nicola Greco, Abdurrahman Ghanem, Dmitrij Zagidulin, Ashraf Aboulnaga, and Tim Berners-Lee. 2016. Solid : a platform for decentralized social applications based on linked data. In https://api.semanticscholar.org/CorpusID:49564404.

[5] Adam Shostack. 2008. Experiences threat modeling at microsoft, (Jan. 2008).

[6] Franziska Von Scherenberg, Malte Hellmeier, and Boris Otto. 2024. Data sovereignty in information systems. *Electronic Markets*, 34, 1, (Feb. 2024). doi:10.1007/s12525-024-00693-4.