Snow: Protocol-Faithful Privacy via Pairwise Device Sync and Pluggable MPC Proving

Nadeem Bhati
nadeemb53@gmail.com

***Abstract*—Privacy has moved from optional to expected. Phones and browsers now ship with encryption by default, and users increasingly demand direct control over their data. In crypto, privacy can be enforced with mathematics rather than policy, yet privacy-focused networks introduce new forms of UX friction: heterogeneous calldata and transaction semantics, client-side proving requirements that strain commodity devices, and brittle multi-device secret handling. Snow addresses these hurdles at the wallet layer by acting as a policy and verification engine that normalizes per-protocol differences, syncs only the minimum state across a user's devices through a locally formed, end-to-end encrypted channel, and routes proving to the most suitable backend—local, Snow-operated REP3, or third-party MPC/coSNARK—while *always* verifying on-device before submission and syncing the private state required for multi-device correctness. Backends remain interchangeable because policy, verification, and device trust are anchored on the client. Snow is chain-agnostic and non-custodial from first principles, and it is designed to let both privacy-native and conventional chains deliver private actions without imposing a UX tax. Benchmarks are reported only when reproduced from public code.**

***Index Terms*—Privacy, Wallets, Zero-Knowledge Proofs, Multi-Party Computation, Pairwise Sync, Authorization**

I. Introduction

Mainstream products now provide encryption by default, but in crypto the core guarantees come from proofs and verification rather than policy agreements. Modern privacy-first networks rely on zero-knowledge proofs (ZKPs) and private state, yet they introduce practical friction in three places: they expose distinct proof stacks and transaction semantics that fragment the wallet UX; they often expect client-side proving on phones or browsers that are ill-suited for sustained workloads; and they make multi-device use brittle by pushing risky secret handling onto users. People want private actions, but setup overhead and performance ceilings stall adoption. Custodial shortcuts reduce friction at the expense of trust.

Snow resolves these issues within the wallet itself by combining deterministic backend selection, local verification of any outsourced result, and short-authentication-string (SAS) verified device sync. The result is a familiar UX that keeps the protocol's privacy guarantees intact.

II. Problem Statement

We focus on four concrete pain points:

- **P1 — Fragmented UX across proof systems.** Protocols use different proof systems, calldata shapes, and key hierarchies, so a single wallet flow breaks unless it adapts at the boundary.
- **P2 — Device-heavy proving and setup.** Many networks implicitly assume client-side proving; phones and browsers struggle, and installing binaries or GPU toolchains adds avoidable friction.
- **P3 — Brittle private state management.** In privacy protocols a user's balance is a client-maintained set of unspent notes/commitments plus viewing keys and Merkle witnesses, not a single on-chain counter. Without a safe sync channel, moving this state between devices is error-prone and incentivizes custodial shortcuts.
- **P4 — Central relays/custody weaken guarantees.** Centralized relays or custodial syncing smooth UX while undermining the very privacy assumptions the protocols are meant to preserve.

III. Design Principles

- **D1 — Protocol-faithful privacy.** Match, and where possible strengthen, each protocol's intended privacy and trust assumptions instead of replacing them with wallet-level policy.
- **D2 — Non-custodial by default.** Users retain control of secrets; no centralized custody is introduced to simplify multi-device use.
- **D3 — Local verification of remote results.** Any outsourced proof is verified on the client before submission so correctness does not depend on provider honesty.
- **D4 — Simplicity at the edge.** Users see one clear flow while adapters encapsulate complexity and protocol-specific details.
- **D5 — Decentralized support paths.** Prefer pairwise encrypted device channels and distributed proving over centralized relay servers.

IV. System Overview

Snow is a wallet-layer policy engine that exposes three interchangeable proving backends—local, Snow-REP3, and third-party MPC/coSNARK—while adapters normalize calldata, witness schemas, proof formats, and authorization rules across protocols. Backend selection is deterministic per circuit and service-level objective so the same action consistently follows the same route.

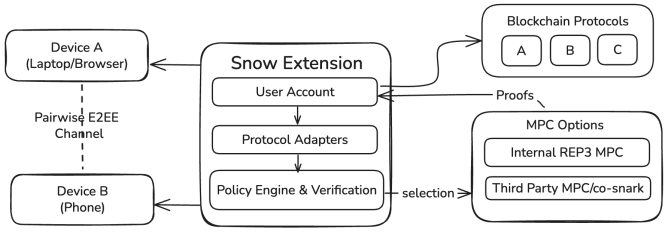

Fig. 1. Architecture: client-side policy and verification, pairwise device channel, and pluggable proving backends.

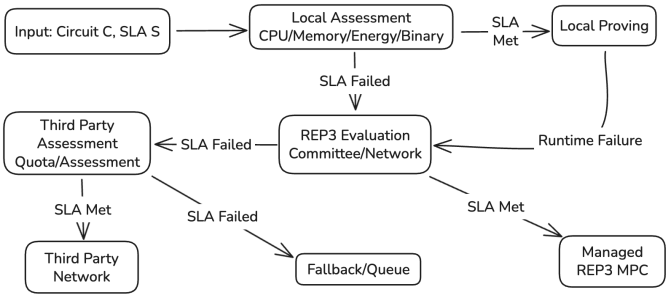

Fig. 2. Backend selection is deterministic and SLA-driven, while verification remains a local invariant.

The Snow Extension manages accounts, adapters, proving policy, MPC jobs, and device pairing. A pairwise device channel provides end-to-end encryption formed locally; security derives from a Noise-XX AKE with SAS binding, independent of the transport. An MPC proving network composed of independent operators computes proofs on secret shares so no single operator learns private inputs. Protocol adapters remain minimal: they encode per-chain calldata, witness schemas, proof formats, and authorization rules, and they operate under client-owned verification.

V. Proving Policy

For a circuit class C and an SLA $\mathcal{S}$, Snow attempts backends in the order Local $\rightarrow$ REP3 $\rightarrow$ Third-Party. The choice is a pure function of $(C, \mathcal{S}, \text{policy\_version})$ so results are predictable across devices. Vouchers and quotas govern metering rather than trust, and correctness is invariant because the wallet verifies any proof locally before it is submitted on-chain.

VI. Pairwise Device-to-Device Secret Transfer

Snow provides secure, non-custodial synchronization of the minimum necessary secrets—such as viewing keys and derived addresses—between a user's authenticated devices using a short-lived, end-to-end encrypted channel formed over a local, untrusted transport. The channel is created locally, the transport may be BLE, Wi-Fi Direct, or public Wi-Fi, and security comes from a Noise XX AKE with a SAS check, transcript binding, and AEAD chunking with counters and explicit revocation. Only derivation and viewing keys, device policy, and adapter metadata are synchronized; root keys move only if the user deliberately executes the same channel flow.

A. Channel Formation

The initiating device displays a QR code or short code that encodes an AKE offer. Snow performs a Noise XX handshake [4] (optionally X3DH [5] where asynchronous initiation is required). The second device scans or enters the code, both sides mutually authenticate, and session keys are derived. No remote relay or mesh is needed for security.

B. Out-of-Band Verification

After pairing, both devices present a short authentication string. The user confirms a match, which prevents relayed or photographed QR attacks. A mismatch aborts the channel and nothing is synchronized.

C. Revocation and Recovery

A user can revoke a lost device from any paired device. Revocation rotates channel keys and tombstones the revoked peer so future traffic remains protected by forward secrecy. Historical ciphertext is accessible only if it was already present at the time of compromise, matching the expectations of ratcheted secure messaging [6]. Recovery remains non-custodial and can involve guardians or printed kits if the user opts in.

D. Mobile/Transport Considerations

On iOS, Bluetooth Low Energy is the default discovery and transport due to platform constraints, while on Android and desktop Snow prefers Wi-Fi Direct for throughput. A local-only WebRTC fallback is available when neither BLE nor Wi-Fi Direct can be negotiated.

E. Provisioning Scope

Snow synchronizes only the minimum client-side state required for protocol-faithful continuity, never custody:

- **Viewing keys (k_v):** Permit each device to independently scan encrypted chain state and discover owned notes.
- **Owned unspent notes/commitments:** The ciphertext/commitment plus the secret fields (value, randomness, address data) necessary to construct a spend.
- **Witness data (Merkle paths):** Inclusion paths for each unspent note to avoid costly full rescans; paths are incrementally updated as new blocks land.
- **Spent nullifiers (own set):** Locally maintained set of nullifiers already revealed by the user to prevent accidental double-spend attempts across devices.
- **Derivation state:** Indices/cursors for stealth/address derivation to prevent collisions and maintain a coherent history.

Root spending keys are *not* synchronized by default. Moving a root key requires the same SAS-verified channel with explicit user action. Device policy and adapter metadata are included to keep behavior identical across devices.

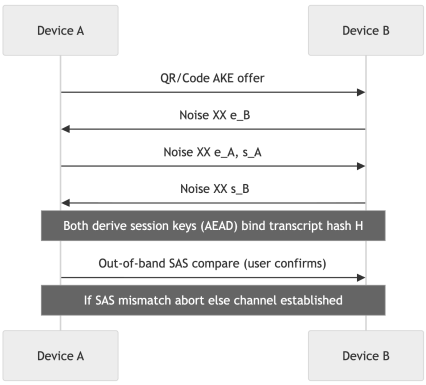

Fig. 3. Pairing flow with SAS verification and transcript binding to prevent relay and reflection attacks.

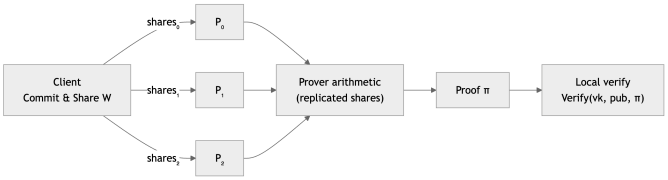

Fig. 4. REP3 pipeline: share the witness, compute over replicated shares, assemble the proof, then verify locally on the client.

F. Example Flow

1) A private payment lands on the desktop. The wallet records a new note, its Merkle path, updates derivation cursors, and—if applicable—marks prior nullifiers as spent.
2) Through the SAS-verified pairwise channel, the desktop syncs *only* the new note, its witness, updated cursors, and nullifier changes to the phone.
3) The phone immediately reflects the correct balance and can construct a valid ZKP spend without rescanning the chain or exporting seeds.

VII. MPC Backends (REP3 / Threshold / coSNARK)

When on-device proving is impractical because of computational cost or setup friction, Snow outsources proof generation to a distributed committee using MPC while keeping private inputs confidential under the chosen security model.

A. REP3 Compatibility Surface

Snow supports replicated sharing over $\mathbb{F}_p$ with one-corruption malicious security in a three-party committee. Deterministic domain separators bind transcripts to public inputs, and the client verifies (π, pub) locally so acceptance does not depend on committee honesty.

B. Threshold/Shamir

For circuits or deployments that value availability and jurisdictional spread over minimal committee size, Snow can route to t-of-n protocols (e.g., SPDZ/VOLE families) using Shamir shares. Claims in this paper refer to REP3 unless stated otherwise.

C. Third-Party MPC/coSNARK

Coordinator-based networks with vouchers or quotas are supported as alternative providers. Vouchers are treated as metering artifacts, and the wallet's local verification remains the acceptance gate.

D. Committee Formation and Backend Selection

Snow maintains a pool of staked operators O with $|O| \geq 3$. For REP3, exactly three operators are sampled per job using a stake-capped VRF lottery with diversity caps to avoid correlated failures. If any selected party cannot meet the SLA for the circuit class, Snow resamples; repeated failures trigger fallback to local proving.

E. Input Provisioning (REP3)

Let $W = (w_1, \ldots, w_m) \in \mathbb{F}_p^m$ be the witness. The client commits to each w_i with $C_i = \mathrm{Com}(w_i; r_i)$ and creates replicated shares using fresh randomness so any two parties reconstruct w_i while any single party learns nothing. Each party receives its share over a Noise-established channel with AEAD, along with the commitments and a per-job transcript identifier. The committee performs consistency checks bound to the commitments before computation. No Shamir polynomials are used in REP3.

F. Computation (REP3)

The parties evaluate the prover arithmetic on replicated shares. Linear operations are local, while multiplications consume preprocessed triples generated during setup. Malicious security against a single corrupted party follows from authenticated transcript checks. The output is a standard proof π with public inputs pub; no party learns W.

G. Verification

The wallet verifies π against pub using the protocol's verification key. A valid proof yields payment to the committee, while invalid or late outputs do not and also affect reputation. This step is identical across backends.

VIII. Security Model

Snow uses distinct models for pairwise sync and MPC proving.

Pairwise Sync: The adversary controls the local network (Dolev–Yao), while Noise XX with SAS, transcript binding, and AEAD counters protect confidentiality and integrity; security reduces to SAS entropy and device integrity.

MPC Proving: In REP3 the system tolerates one corrupted party in a three-party committee; an adversarial majority can harm liveness but not correctness because the wallet verifies proofs locally.

A. Threat Model: Pairwise Device Sync

We assume the adversary controls the local transport (BLE/Wi-Fi/WebRTC) and can relay offers; SAS comparison binds the AKE transcript to the human check, preventing QR/photo relay. Syncing notes/witnesses does not weaken spend correctness because proof verification and nullifier checks occur locally.

B. Threat Model: MPC Proving (REP3)

The adversary may control at most one operator in the three-party committee and can deviate arbitrarily. Goals are (1) input privacy, where any single party's view is statistically independent of W, and (2) proof integrity, where either a valid π is produced and verified or no output is accepted. Assumptions include the hardness underpinning the MPC and ZKP schemes, at least two honest parties, correct input provisioning, and local verification. An adversarial majority can at most delay or deny service.

C. Security Analysis

Device Compromise. Hardware-backed key storage, optional spending limits, and rapid revocation bound the blast radius. Compromising one device does not reveal secrets that were never synchronized to it.

Proof Integrity. Because the wallet verifies π locally, correctness is independent of committee behavior; dishonest operators affect liveness, not acceptance.

IX. Governance and Economics

Snow operates an open provider registry with bonds and client-side sampling. A stake-capped VRF with jurisdictional and infrastructure diversity caps selects committees, and bonds are slashable for equivocation, SLA violations, or invalid transcripts. Both coordinatorless and coordinator-based modes are supported; vouchers, where present, are treated strictly as quota tokens. For cold start, Snow may bootstrap with an audited allowlist of operators and decentralize over time.

A. Sampling

Given the operator set O, Snow samples a REP3 committee $\{P_0, P_1, P_2\} \subset O$ per job using VRF proofs and diversity caps. Selection metadata is included in a client-signed job envelope. The client establishes Noise XX channels and provisions replicated shares. If a selected party fails to meet the SLA (e.g., p95 latency bound for circuit class C), Snow resamples; repeated failures trigger local fallback.

B. Fees, Reputation, Slashing

Jobs are priced via a posted-price menu with sealed-bid support that converges toward a base-fee market as utilization stabilizes. Operators are paid per valid proof, latency percentiles feed reputation, and bonds are slashable for equivocation, failure to co-sign within SLA, or invalid transcripts.

TABLE I
End-to-end metrics (PoC harness)

Metric	Result
MPC proving latency (E2E)	**5.7 s**
Local verification (client)	**154 ms**
Proof system	UltraHonk
Proof size	456 field elements ($\approx$ 14,592 bytes)
Security model	Malicious-secure REP3 (1-of-3)

X. Benchmarking Methodology

End-to-end wall-clock measurements cover encode/share, network transit, MPC computation, proof assembly, and client verification. Comparisons across backends use identical circuits and a dockerized harness with fixed seeds and AEAD transport. Scripts are published so the setup can be reproduced.

XI. Benchmark Setup

Codebase: https://github.com/getsnowxyz/aztec-mpc-poc

Circuit: Aztec nullifier (UltraHonk)

MPC Protocol: REP3 (3-party, malicious-secure)

Scope: PoC harness; we do not state host hardware or RTT claims here.

XII. Performance Results

A. Aztec Nullifier (UltraHonk, REP3, 3-Party)

B. Integration Validation

Protocol adapters satisfy chain-specific requirements while maintaining a unified UX, and the stated protocol fidelity and performance targets hold for the scoped circuits and backend.

XIII. Related Work and Positioning

Single-stack wallets do not generalize across proof systems, and while MPC/coSNARK networks commoditize off-device proving, they do not by themselves standardize policy, device trust, or provider selection. Snow anchors these concerns on the client and routes to local, Snow-REP3, or third-party backends through chain-agnostic adapters. Custodial sync services improve convenience at the cost of non-custodial guarantees; Snow achieves the same convenience with SAS-verified pairwise sync and client-side verification. Recent progress in collaborative proving validates outsourcing as a viable path; Snow supplies the wallet abstraction that preserves protocol intent.

XIV. Conclusion

Snow removes practical friction without diluting protocol guarantees by abstracting per-protocol differences, synchronizing only the minimum state through locally formed pairwise channels, and outsourcing proving when local paths are impractical while always verifying proofs on-device. The approach scales across privacy networks and remains non-custodial by construction.

References

[1] A. Shamir, "How to Share a Secret," *Communications of the ACM*, 1979.

[2] I. Damgård et al., "Practical Covertly Secure MPC with Dishonest Majority," *CRYPTO*, 2012.

[3] T. Araki et al., "High-Throughput Secure Three-Party Computation for Malicious Adversaries and an Honest Majority," *ACM CCS*, 2016.

[4] T. Perrin, "The Noise Protocol Framework," 2018.

[5] M. Marlinspike and T. Perrin, "The X3DH Key Agreement Protocol," Signal, 2016.

[6] M. Marlinspike and T. Perrin, "The Double Ratchet Algorithm," Signal, 2016.

[7] A. Gabizon, Z. J. Williamson, O. Ciobotaru, "PLONK," 2019.

[8] P. Boyle et al., "Efficient Two-Round OT Extension and VOLE," 2021.