

Decentralized Platforms: Comprehensive Comparative Analysis

Section 4 – Platform Deep Profile: Peergos
Version 3.0 | 16 August 2026 | Segment 04

4.1 Overview

Attribute	Value
Full name	Peergos
Type	P2P encrypted global filesystem + social + application platform
Primary identity	Self-authenticating cryptographic identity (portable)
Core primitive	Content-addressed encrypted blocks + cryptographic capabilities
Official site	https://peergos.org
Documentation	https://book.peergos.org
Source	https://github.com/Peergos
Primary design goal	Private, user-controlled space with metadata protection + sandboxed apps

Peergos is a **vertical platform**: protocol + reference server + official clients + social layer + permissioned application runtime. This produces the strongest integrated privacy model in this report, at the cost of a very small independent client ecosystem.

4.2 History and Design Philosophy

Peergos has been under development for more than a decade. Name from Greek Πύργος (stronghold). Core principles: only the user reads their data/metadata/friend list; servers never see plaintext; identity and data portable; apps sandboxed with explicit permissions; quantum resistance where practical.

2025 milestones: Peergos 1.0, native desktop apps, full Android app, high-performance sync, post-quantum hybrid sharing (Curve25519 + ML-KEM-1024), 20 releases, tripled paying customers. **2026 direction:** polish, iOS, deeper OS integration; further PQ work on identity (SLH-DSA) and writer keys (ML-DSA hybrid).

4.3 Architecture

Every write is signed. Files are chunked (~5 MiB), padded, and encrypted client-side. Two random symmetric keys per file/directory (not convergent). Server stores only ciphertext and cannot deduce sizes or structure. Clients verify hashes of every block.

4.4 Cryptography and Privacy

Unshared data: already quantum-resistant (hashing + symmetric only). **Sharing:** hybrid X25519 + ML-KEM-1024 (deployed 2025) protects against harvest-now-decrypt-later. **Metadata:** filenames, sizes, directory structure, times, types, and friend list designed not to be exposed to the server. Independent audits by Cure53 and Radically Open Security.

4.5 Capability Model

```
Capability { owner: PublicKey, writer: PublicKey (or derived), mapKey: SymmetricKey, location: ContentAddressedPointer, access: Read | Write | ... }
```

Access control is capability-based and enforced client-side. Capabilities can be granted, attenuated, and revoked cryptographically. The server never evaluates plaintext ACLs. This is a major architectural difference from server-mediated ACL systems.

4.6 Application Platform

Sandboxed HTML/JS/CSS apps. Anatomy: assets/ (with index.html), optional data/, mandatory peergos-app.json.

Minimal manifest:

```
{ "displayName": "App", "description": "does something", "launchable": true }
```

Richer example:

```
{ "displayName": "Forum", "version": "1.0.1", "author": "peergos", "launchable": true, "permissions": ["EXCHANGE_MESSAGES_WITH_FRIENDS", "ACCESS_PROFILE_PHOTO" ] }
```

Key permissions: STORE_APP_DATA, EDIT_CHOSEN_FILE, READ_CHOSEN_FOLDER, EXCHANGE_MESSAGES_WITH_FRIENDS, ACCESS_PROFILE_PHOTO. Apps run under CSP isolation. REST-like API under /peergos-api/v0/.

4.7 Clients and Interfaces (mid-2026)

Almost entirely official: Web UI, native desktop (Win/macOS/Linux/Flatpak), Android, CLI, FUSE, WebDAV. iOS planned for 2026. Independent third-party clients essentially non-existent. Extension occurs via sandboxed apps inside Peergos, not alternative protocol clients.

4.8 Sync Engine & Self-Hosting

Bi-directional folder sync with optional deletes; only changed chunks transferred; limited write capability so device compromise does not expose the rest of the user's data. Fully self-hostable (2 cores / 2 GiB recommended); local or S3 block store; SQLite sufficient for thousands of users. Server sees only ciphertext.

4.9 Current Status (August 2026)

1.x in production; native desktop + Android shipped; PQ hybrid sharing deployed; active development; growing paying base; iOS and further PQ migration in progress; client ecosystem remains vertically integrated by design.

4.10 Strengths and Limitations

Strengths: strongest combined privacy model for storage+sharing+social; PQ sharing deployed; metadata protection first-class; capability-based access; audited; self-hostable with low host trust; sandboxed apps; portable identity; practical limited-capability sync.

Limitations: almost no independent clients; smaller network effect; higher cost to write alternative clients; innovation concentrated in official stack.

4.11 Comparative Position

Dimension	vs Nostr	vs Solid	vs IPFS	vs Matrix
Default encryption of stored data	Far stronger	Stronger	Far stronger	Stronger for files
Client diversity	Far lower	Lower	Far lower	Far lower
Access control	Capabilities	ACL/WebID	None native	Server + E2EE
Quantum-resistant sharing	Yes	No	No	No
Sandboxed apps	Yes	Possible	No	Limited
Identity portability	High	High	N/A	Medium

Peergos is the clearest example of a vertical, capability-oriented, privacy-first design. It optimizes for confidentiality, metadata resistance and integrated applications at the expense of the broad client experimentation seen in pure protocol systems.

4.12 Key Primary Sources

- <https://peergos.org> • <https://book.peergos.org> • <https://github.com/Peergos>
- <https://github.com/Peergos/example-apps> • <https://peergos.org/posts/2025>
- Features, cryptography and audit pages on the official site