Research Document on the Development of Zero Mail: An Encrypted, P2P, 100% Client-Side Storage Mail System for Desktop

Abstract

Zero Mail is conceptualized as a fully decentralized, encrypted peer-to-peer (P2P) email system for desktop environments, emphasizing 100% client-side storage, end-to-end encryption (E2EE), and no reliance on central servers. This research synthesizes feasibility assessments, existing projects, technical challenges, solutions, and implementation strategies based on ongoing developments in privacy-focused P2P technologies as of February 2026. While pure client-side storage poses significant hurdles for asynchronous delivery, viable approaches leverage distributed P2P networks (e.g., IPFS, Tor) for temporary encrypted message queuing. Key insights draw from projects like Eppie, Peergos, Quiet, BitMessage, and Ricochet Refresh, highlighting trade-offs in usability, performance, and adoption. The document concludes with a roadmap for development, underscoring the system's potential for censorship-resistant, privacy-preserving communication.

Introduction

Traditional email relies on centralized servers (e.g., SMTP/IMAP), exposing users to surveillance, data breaches, and single points of failure. Zero Mail aims to address this by creating a desktop-only system where:

- **Encryption**: All messages are E2EE using asymmetric cryptography (e.g., elliptic-curve keys).
- **P2P Architecture**: Direct peer connections for online delivery; distributed storage for offline scenarios.
- **Client-Side Storage**: All data (inbox, sent, drafts) resides locally on the user's device, encrypted at rest.
- **Desktop Focus**: Cross-platform apps (Windows, macOS, Linux) prioritizing security over mobility.

As of 2026, advancements in P2P protocols (e.g., libp2p, Tor) and post-quantum cryptography make this feasible, though strict "100% client-side" limits async functionality without compromises like temporary peer-hosted queues. This document builds on prior analyses, incorporating updates from 2025-2026 to guide development.

Related Work

P2P encrypted messaging has evolved, but true email-like systems remain niche due to offline delivery challenges. Below are key projects with 2026 status updates:

Project	Description	Key Features	2025-2026 Updates	Status & Limitations

Eppie	Open-protocol P2P email with decentralized identity; bridges to traditional email (Gmail, Proton). E2EE via elliptic-curve crypto; IPFS/SBBS for storage/transport. Desktop app (C#/Uno).	Providerless accounts (BIP39 seeds), crypto addresses as inboxes (Bitcoin/Ethereum), HTML rendering, PGP support.	Testnet live with decentralized addresses; features added: HTML display, email validation, new settings (Oct 2025 release v1.0.185-preview). Crypto inbox integration (e.g., Gmail to Bitcoin addresses). Active GitHub (last commit Jan 2026). Public Testnet planned.	In preview; not fully decentralized yet (some cloud nodes); cross-platform expansion ongoing.
Peergos	P2P encrypted filesystem/messenger/email bridge. E2EE (TweetNaCl; post-quantum for unshared data). IPFS-based storage; self-hosting.	Merkle-CHAMP structure for chunks, secret links, fine-grained access. Supports email-like sharing.	2025: Native apps, post-quantum sharing (MLKEM-1024 hybrid), server migration, P2P HTTP reliability fixes. Quic transport (Dec 2025). 2026 plans: Usability polish, iOS app, deeper OS integration. Active (releases Nov/Dec 2025).	Email is a bridge, not core; focuses on storage/social over pure mail.

Quiet	Encrypted P2P team chat (Slack alternative) on Tor/IPFS. Local storage; async channels.	Metadata hiding via Tor, timed deletion, notifications. Desktop/mobile apps.	Mobile release v6.5.1 (Feb 2026). F-Droid support incoming. Avatars/DMs soon. Active GitHub (1,924 releases).	Chat-focused; no native email but shows P2P async viability for groups.
BitMessage	P2P encrypted "email" with broadcast/flooding; PoW anti-spam. Temporary node storage (~2 days).	Anonymity via pubkey-derived addresses; trustless. Python/C++ desktop client.	No updates since emergency patch (v0.6.3.2, Feb 2018). Inactive/archived; last commit Nov 2025 (lint fix, but tied to v0.6).	Broadcast inefficiency; low adoption; vulnerable to old exploits if unpatched.
Ricochet Refresh	P2P IM via Tor hidden services; E2EE, metadata-free.	Direct connections; .onion IDs. Desktop app.	No 2025-2026 updates found; last major activity pre-2022. GitHub inaccessible (404). Possible archival.	Real-time only; no offline queuing; IM over email.

Other trends: Centralized encrypted emails (ProtonMail, Tuta) dominate 2026 market, with AI integrations and post-quantum focus, but lack true P2P. No new pure P2P email breakthroughs in 2025-2026; emphasis on hybrids.

Technical Feasibility

Zero Mail is viable using modern stacks:

- **Encryption**: Libsodium or age for E2EE; pubkey-derived addresses. Add post-quantum (e.g., MLKEM) for future-proofing.
- **P2P Layer**: Libp2p for discovery/connectivity; Tor for anonymity/NAT traversal. IPFS for distributed temp storage (encrypted chunks with TTL).
- **Storage**: Local SQLite (encrypted via SQLCipher); no permanent network storage to meet "100% client-side."
- **Delivery**: Direct push if online; sender queues/retries or IPFS pinning for offline (peers hold encrypted blobs temporarily).
- **Desktop Implementation**: Rust/Go core + Tauri UI for cross-platform efficiency.

Proof-of-concept from Eppie/Peergos shows scalability for small-medium networks.

Challenges and Solutions

Complexity

Intricate integration of P2P, crypto, and UI.

- **Solutions**: Modular libs (libp2p, OpenDHT); MVP start (online-only); IBC for simplified keys.

Usability (Discovery/Offline)

Manual key exchange; offline delivery unreliable without storage.

- **Solutions**: DHT for discovery; relay nodes/IPFS for queuing; tutorials/dashboards. Hybrid bridges to SMTP for onboarding.

Performance

High bandwidth/latency from churn/retries.

- **Solutions**: RLNC coding; quic/UDP optimizations; caching/load balancing.

Limited Adoption

Network effects hinder growth.

- **Solutions**: Marketing (social/influencers); incentives (crypto rewards); niches (privacy communities); bridges to legacy email.

Additional: Spam via PoW/reputation; security audits essential.

Implementation Path

1. **Core Prototype**: Rust lib for encryption/P2P (libp2p + Tor arti); local DB.
2. **MVP Features**: Online delivery, key exchange, UI (Tauri).
3. **Enhancements**: Offline via IPFS; bridges (Eppie-style); post-quantum.
4. **Testing/Deployment**: Open-source on GitHub; beta with privacy groups.
5. **Timeline**: 6-12 months for MVP, iterating on Peergos/Quiet models.

Future Considerations

Monitor quantum threats; integrate MLS for groups. Expand to mobile if demand grows. Regulatory risks (e.g., encryption bans) minimal for P2P.

Conclusion

Zero Mail represents a step toward sovereign communication, leveraging 2026 P2P advancements. While challenges persist, solutions from active projects like Eppie and Peergos provide a blueprint. Development should prioritize security audits and user feedback for sustainable adoption.

References

- Citations drawn from web searches on P2P systems (2025-2026 updates).
- Key sources: Eppie GitHub/docs, Peergos blog/releases, Quiet repo, BitMessage archive.