Obsidian Wallet by Zpoken | Grant Proposal

Contact Details:

- Email:

mike.yezhov@zpoken.io

Summary

Zpoken proposes to develop Obsidian Wallet

, a user-centric, privacy-focused wallet for the Aztec network. This initiative aims to accelerate privacy adoption and foster ecosystem growth. Obsidian Wallet will offer essential features including account management, transaction capabilities, and asset handling with support for both private and public transactions on the Aztec network. To further catalyze ecosystem growth, Obsidian Wallet will include a developer-friendly SDK enabling seamless integration with other applications and encouraging the creation of privacy-enhanced solutions across various use cases.

Estimated Start and End Dates

- Start Date:

October 7, 2024

- Testnet Release Date:

December 27, 2024

- Further Releases:

February 10, 2025

About Us

Zpoken is a cryptography research and blockchain engineering organization that contributes to various web3 protocols and ecosystems (like Near, Wormhole, Penumbra, =nil; and others). For this grant proposal, the main relevant experience includes:

- Penumbra:

2 years of contributing to this ecosystem, building a robust browser-based wallet (Prax extension was initially hosted on Zpoken's account and transferred to the Penumbra team later). Penumbra Web | Prax Wallet

- Near Privacy Initiative:

Working in close collaboration with other teams and Near Foundation on extending NEP-141 token with a darkpool feature and the NEAR ZK light client. NEAR ZK Light Client

- Metamask Snap for =nil;:

Created a snap with support for contract wallets for the =nil; ecosystem.nil Snap | Tweet

We have an in-depth understanding of privacy technologies and account abstraction.

Technical Architecture

We plan to implement Obsidian Wallet as a browser extension with the following stack:

- Front-End:

React, Vite, Typescript, Zustand, Shadcn/ui, Tailwind for a responsive web application providing functionalities like account creation, import, transactions, and asset viewing.

- SDK:

For developers to integrate wallet functionalities into their applications.

- Offscreen Workers:

Experience with offscreen workers for Chrome extensions to build proofs for transactions in parallel or sync up the state, bypassing extension service worker limits. Example

Features

- Wallet Integration with Wallet Connect
- Account Management:

Support account creation, import, syncing, backups, and migrations.

- Transaction Functionality:

Support both public and private transactions on the Aztec network, providing a user-friendly transaction interface.

- Asset Flow:

Display balance changes in a user-friendly format.

- Asset Viewing:

Display users' public and private asset balances, provide transaction history.

- PXE Integration:

Directly call the PXE library provided by Aztec to interact and handle the simulation and proof generation of private transactions.

- Built-in Asset Registry:

Assets metadata and icons stored locally in the extension to prevent privacy leaks.

- Built-in Price Indexer:

Scanning blocks and storing the latest prices in the extension storage to display equivalent prices; using centralized price providers can lead to privacy leaks.

Design preview

Below is the first iteration of wallet interfaces:

[

obsidian_main

1600×2880 198 KB

](https://europe1.discourse-cdn.com/flex013/uploads/aztec/original/2X/3/3d529daa2c6d80700c750e0bf4437a0a9e33b8a9.jpeg)

[

obsidian_create

1600×2880 285 KB

](https://europe1.discourse-cdn.com/flex013/uploads/aztec/original/2X/8/80899b4a653c75ad70311380b7b32e9ea29485c9.jpeg)

[

obsidian_create1

1600×2880 168 KB

](https://europe1.discourse-cdn.com/flex013/uploads/aztec/original/2X/0/0c58b606c376d87033c49df82ee76cddaa383100.jpeg)

[

obsidian_main1

1600×2880 242 KB

](https://europe1.discourse-cdn.com/flex013/uploads/aztec/original/2X/2/212493de10704d2a8e05a77582539f7a66ca46e8.jpeg)

[

obsidian_main1_light

1600×2880 240 KB

](https://europe1.discourse-cdn.com/flex013/uploads/aztec/original/2X/9/9dd4b6170510ea8fde87498b88d082c04a2ff99e.jpeg)

[

asset

1600×2880 261 KB

](https://europe1.discourse-cdn.com/flex013/uploads/aztec/original/2X/f/f7eb0951722c62e8864851391e5f3a2ec9b65166.jpeg)

Phase Breakdown (Weeks 1-12)

Phase 1: Project Setup and Requirements Finalization (Weeks 1-2)

- Deliverables:
- Finalized project requirements and technical specifications for the Chrome extension wallet.
- Design architecture for the extension focusing on core wallet functionality, security, and interaction with the Aztec network.
- Definition of key management strategy, contract interactions, and onboarding flows.
- Setup of project repositories, tools, and development environment.
- Finalized project requirements and technical specifications for the Chrome extension wallet.
- Design architecture for the extension focusing on core wallet functionality, security, and interaction with the Aztec network.
- Definition of key management strategy, contract interactions, and onboarding flows.
- Setup of project repositories, tools, and development environment.
- Key Activities:
- Team setup, role assignments, and technical research on Chrome extension development and Aztec protocol.
- Define the architecture for key components: onboarding, contract management, and secure key storage.
- Design the user interface (UI) and user experience (UX) for the Chrome extension.
- Establish project milestones and development timelines.
- Team setup, role assignments, and technical research on Chrome extension development and Aztec protocol.
- Define the architecture for key components: onboarding, contract management, and secure key storage.
- Design the user interface (UI) and user experience (UX) for the Chrome extension.
- Establish project milestones and development timelines.

Phase 2: Core Wallet Infrastructure & Key Management (Weeks 3-4)

- Deliverables:
- Foundation of the Chrome extension with basic wallet functionality.
- Implementation of keypair generation (signing keypair, nullifier keypair, incoming and outgoing viewing keypairs).
- Secure key storage within the extension using encrypted local storage.

- Initial user interface for account management and key viewing.
- Foundation of the Chrome extension with basic wallet functionality.
- Implementation of keypair generation (signing keypair, nullifier keypair, incoming and outgoing viewing keypairs).
- Secure key storage within the extension using encrypted local storage.
- Initial user interface for account management and key viewing.
- Key Activities:
- Implement core wallet functionality including key generation for signing, viewing, and privacy purposes.
- Develop a secure key storage system using encryption for sensitive data.
- Build UI elements for creating, viewing, and managing keys in the wallet.
- Begin basic testing to ensure proper key generation and storage.
- Implement core wallet functionality including key generation for signing, viewing, and privacy purposes.
- Develop a secure key storage system using encryption for sensitive data.
- Build UI elements for creating, viewing, and managing keys in the wallet.
- Begin basic testing to ensure proper key generation and storage.

Phase 3: User Onboarding & Account Creation (Weeks 5-6)

- Deliverables:
- User-friendly onboarding process that generates keys automatically behind the scenes.
- Passkey authentication support to replace the traditional 12-word seed phrase.
- Account creation with automatic contract generation and delayed deployment (when required).
- User interface for onboarding with minimal steps and a seamless flow.
- User-friendly onboarding process that generates keys automatically behind the scenes.
- Passkey authentication support to replace the traditional 12-word seed phrase.
- Account creation with automatic contract generation and delayed deployment (when required).
- User interface for onboarding with minimal steps and a seamless flow.
- Key Activities:
- Implement passkey authentication or other user-friendly mechanisms to replace seed phrases.
- Develop a streamlined onboarding process that creates the user’s keys and account without user intervention.
- Set up contract deployment logic to allow for delayed deployment (i.e., contracts only deployed when the user initiates their first transaction).
- Test onboarding process to ensure minimal user friction and successful contract setup.
- Implement passkey authentication or other user-friendly mechanisms to replace seed phrases.
- Develop a streamlined onboarding process that creates the user’s keys and account without user intervention.
- Set up contract deployment logic to allow for delayed deployment (i.e., contracts only deployed when the user initiates their first transaction).
- Test onboarding process to ensure minimal user friction and successful contract setup.

Phase 4: Transaction Management & (Weeks 7-8)

- Deliverables:
- Implementation of transaction batching and private/public transaction handling.

- UI to display private and public token balances, transaction history, and key data.
- Security features for transaction signing and private authorization (authwits).
- Implementation of transaction batching and private/public transaction handling.
- UI to display private and public token balances, transaction history, and key data.
- Security features for transaction signing and private authorization (authwits).
- Key Activities:
- Build functionality to handle transaction batching, allowing multiple actions to be submitted in one transaction.
- Develop user interface elements for managing private and public balances, transaction history, and pending transactions.
- Security testing to ensure proper handling of private transactions and authorization witnesses (authwits).
- Build functionality to handle transaction batching, allowing multiple actions to be submitted in one transaction.
- Develop user interface elements for managing private and public balances, transaction history, and pending transactions.
- Security testing to ensure proper handling of private transactions and authorization witnesses (authwits).

Phase 5: Token Bridge, Fee Management & Advanced Contract Features (Weeks 9-10)

- Deliverables:
- Integration of Aztec’s Token Bridge (Asset Portal) to support cross-chain token transfers.
- Fee management system that allows users to pay fees using Aztec’s native “fee juice” or custom paymasters.
- Implementation of batch transaction capabilities, enabling users to submit multiple actions in one go.
- Interface for selecting different fee payment options based on user token balances.
- Integration of Aztec’s Token Bridge (Asset Portal) to support cross-chain token transfers.
- Fee management system that allows users to pay fees using Aztec’s native “fee juice” or custom paymasters.
- Implementation of batch transaction capabilities, enabling users to submit multiple actions in one go.
- Interface for selecting different fee payment options based on user token balances.
- Key Activities:
- Develop integration with Aztec’s Token Bridge to allow for seamless token transfers between Aztec and other networks.
- Implement fee management functionality showing users fee payment options and suggestions based on available tokens.
- Test and optimize token bridging and fee payment functionality.
- Build advanced contract features such as transaction batching and smart contract interactions.
- Develop integration with Aztec’s Token Bridge to allow for seamless token transfers between Aztec and other networks.
- Implement fee management functionality showing users fee payment options and suggestions based on available tokens.
- Test and optimize token bridging and fee payment functionality.
- Build advanced contract features such as transaction batching and smart contract interactions.

Phase 6: Privacy, Security, and Backup/Migration Features (Weeks 11-12)

- Deliverables:
- Strong privacy mechanisms to protect user data and prevent potential PXE data leaks.

- Secure connection with Aztec PXE (Privacy Execution Environment) to prevent unauthorized access.
- Account backup and migration features allowing users to export/import accounts between devices.
- Final extension security audits, optimization, and deployment readiness.
- Strong privacy mechanisms to protect user data and prevent potential PXE data leaks.
- Secure connection with Aztec PXE (Privacy Execution Environment) to prevent unauthorized access.
- Account backup and migration features allowing users to export/import accounts between devices.
- Final extension security audits, optimization, and deployment readiness.
- Key Activities:
- Implement secure communication with Aztec PXE, ensuring no unauthorized access or data leaks.
- Develop features for users to back up and migrate their accounts (keys, transaction history, notes) securely.
- Conduct extensive privacy and security testing focusing on encryption, key management, and contract interaction.
- Perform final tests and code optimization, followed by a security audit of the extension.
- Implement secure communication with Aztec PXE, ensuring no unauthorized access or data leaks.
- Develop features for users to back up and migrate their accounts (keys, transaction history, notes) securely.
- Conduct extensive privacy and security testing focusing on encryption, key management, and contract interaction.
- Perform final tests and code optimization, followed by a security audit of the extension.

Future Plans

After Testnet Release, we plan to ship the following features:

- Built-in Asset Registry:

Assets metadata and icons to be stored locally in the extension.

- Built-in Price Indexer:

Scanning blocks and storing the latest prices in the extension storage.

- Monolithic Repository:

Create a monolithic repository of Aztec web code (a monorepo) to simplify work and make broad cross-package changes more feasible.

Grant Amount Requested: $90,000

Budget Breakdown

Resource

Unit Price

Quantity

Amount

Blockchain Engineer

$1750/week

12 weeks

$21,000

Blockchain Engineer

$1750/week

12 weeks

$21,000

Front-End Engineer

$1750/week

12 weeks

$21,000

Designer

$1500/week

3 weeks

$4,500

Further Development and Support Funds

-

-

$20,500

Total

$88,000

Grant Budget Rationale

- Team Costs:

To cover the costs of engineers and designer to deliver the product.

- Further Development and Support Funds:

Funds allocated for post-release works related to product improvement, new features development, bug fixing, and support.

Questions and Requests

We have experience implementing the Penumbra View Service as a separate component within browser extensions. This service shares a similar architecture with PXE: it scans blocks, decrypts notes, detects nullifiers and their spending, and stores the data in a local database. We utilized WebAssembly (WASM) for high-performance scanning and IndexedDB for browser-based data storage, incorporating migration mechanisms and a permissive system for data access. Moreover, we've abstracted the storage layer to allow for alternative storage methods in environments where IndexedDB isn't available. Our browser-based implementation of PXE can be packaged as a separate module, making it easily integrable into other wallets.

We look forward to collaborating with the Aztec team in shipping Obsidian Wallet for user onboarding, on-chain privacy adoption, and promotion of the Aztec ecosystem.