[

Obsidion Wallet

3973×1273 206 KB

](https://europe1.discourse-cdn.com/flex013/uploads/aztec/original/2X/4/423dac5a8d8a1e637b2c4f20d5e1b4745c941498.jpeg)

Contact details

Telegram: @zkMike @madztheo @porco_rosso_j

Summary

Obsidion Wallet

is a fully-featured wallet designed for seamless interaction with dapps and protocols on the Aztec Network.

Our vision is to reimagine the user experience to make Web3 as intuitive and accessible as Web2, enabling actions like sending funds to an email address rather than a hexadecimal one.

Initially our wallet will be developed as a browser webapp, with the goal of it eventually becoming a Chrome extension and potentially later on a desktop app with a passthrough Chrome extension.

Details

Accounts

Sign in with Apple

Users can create an Aztec account using Sign in with Apple

, which verifies ownership of their email through a JWT. Their Aztec account's public key is then associated with their email address in the Aztec Account Email Registry.

This allows anyone on Aztec to send tokens to or interact with others in a private way using simply their email address!

Sign in with Google

Similarly, users can create an account using Sign in with Google

, associating their Google email with their Aztec account.

Passkey-based Accounts

When a user creates an Aztec account, they can use a passkey (via WebAuthn) to derive an Aztec master key, which in turn is used to derive the signing key, viewing key, nullifier key etc.

Aztec Account Email Registry

A smart contract with a public mapping of email addresses to Aztec public keys.

Users can associate their email address with their public key by providing a valid proof of email ownership to the registry smart contract.

Send to Email

Users can send tokens directly to an email address. If the recipient's Aztec account is linked to their email, the transfer is seamless. Otherwise, the recipient receives an email with a unique salt and instructions to claim the tokens. The note is encrypted using H(email + salt)

, and the recipient must prove (via a zkEmail / proof of email) that they own the email to be able to redeem the note.

Tokens

Balances

Viewing private and public token balances.

Transfers

Sending tokens to recipients via Aztec address or email address.

When sending tokens the user will have the option to hide sender

as well as provide a message or reference number.

For tokens with compliance requirements, zkPassport compliance proofs will be automatically generated.

History

Transaction history.

Trusted Token Registry

A registry of well known token contract addresses and their associated logos.

Account Abstraction

Authentication Modules and Multi-sig

We take modular approach to enable signature abstraction for account contract: delegating tx signature validation to external authenticator contracts (also storing keys) for each signing method. This keeps the account contract itself as minimal as possible while enabling customizability in validation, supporting various methods, e.g. ECDSA K256/P256, Schnorr, zk-email, social logins, etc… (PoC)

Also, inspired by ArgentX, the most popular browser extension wallet on StarkNet, andSafe(Gnosis), Obsidion plans to provide multi-sig feature as an authenticator module, i.e. multi-sig authenticator. It allows users to add multiple accounts as co-signers, essentially making it multi-factor multi-sig account, e.g. 2/2 multi-sig with passkey account as primary signer and zk-email account as a 2FA signer.

Additionally, this modular architecture allows other types of extensional helper modules to be installed on the account to enhance security and UX further, such as spend limit, recovery, and automation modules. With shared interfaces between account and each type of module, it's possible to establish a standard eventually so that external wallet/dapp developers can develop their own custom modules to build their solutions on top of our account stack.

Module Examples:

- Zodiac Module for Gnosis Safe
- ERC7579 Modules

Nonce Management

To maximize the benefit of nonce abstraction, our account implements two-dimensional nonce enabling parallel transactions like you see in ERC4337 space, e.g. ZeroDev's Parallel UserOps.

Fee sponsorship

As the RFP requirement suggests, to facilitate smooth onboarding, we gas-sponsor the users' first transaction: account deployment and/or the first execution that runs in public context. Also, a Fee Payment Contract helps users pay fees in their preferable assets (privately if possible).

Transaction Previews

When interacting with a dapp, users are shown a clear, user-friendly summary of the transaction's effects. This ensures users have a complete understanding of the consequences of their actions before proceeding.

PXE runtime

PXE should be run on the client side to offer maximum privacy to the user and prevent any data leaks that could compromise the privacy-preserving features of Aztec. But as of today, PXE cannot run in a browser environment and only in Node.js, thus making it harder to keep this promise entirely. In the meantime, as PXE support for browser is expected to arrive a bit later, we plan on providing two solutions so users can use our wallet with different levels of assumptions and convenience

The first option, which will likely be the default, will be to connect to a dedicated Node.js server hosted by us to run PXE and encrypt all communications between the user device and the server. The second option will be to let the user run its own docker instance of PXE on their machine so they don't have to rely on a centralised server (even if with encrypted communication) to use our wallet. This will be available in the settings and as an option during the onboarding flow.

Encrypted Blob Storage

Snapshot Sync

Encrypted notes and other private PXE state can be stored via our encrypted blob storage service, allowing users to synchronise their wallets across devices quickly and easily without the need for a full rescan of encrypted notes etc.

Address Book

The address book can also be persisted in our encrypted blob storage, ensuring it is accessible and usable across all of a user's devices.

Identity Proofs

zkPassport is integrated to provide first-class identity management in the wallet for seamless integration with dapps and protocols, allowing automatic identity proof generation.

For example, after a user grants the appropriate passport permissions to a stablecoin token contract to access their zkPassport, the wallet handles the generation of compliance proofs going forward whenever the token is transferred. This could involve generating a proof that the person is not from a sanctioned country and that they are not in the OFAC SDN (or country-specific equivalent) list.

Other interesting use cases for identity proofs: RWA, DAO Sybil identifier logarithmic-weighted voting.

Incentivising Adoption

We can incentivise adoption by paying users X USDC to invite their friends to try out Obsidion Wallet. Their friends will receive an email inviting them to download and try out the wallet to claim X USDC that's waiting for them.

Future Work

- Mobile version with more seamless zkPassport integration and Face ID
- Phone to phone transfer through NFC (akin to Apple Tap to Cash)
- Send token via a link (akin toPeanut)
- private payment app using Peanut & Railgun
- private payment app using Peanut & Railgun

Estimated Start and End Date

Start date

: 8th Oct

End date:

31st Dec

Duration:

12 weeks

Our Team

We are a team of 3 software engineers with experience in the blockchain and ZK space and a track record of successfully shipping products.

Grant Milestones and Roadmap

October

- Initial barebones wallet functionality
- Wallet skeleton and PXE interaction
- Wallet skeleton and PXE interaction
- Onboarding account creation flow
- Key generation and management
- Key generation and management
- Token balances and transfers

November

- zkPassport integration
- zkPassport compliance proof generation for compliant stablecoin tokens

December

- Email address to Aztec account association via zkEmail proof of email
- User testing and product polishing

Grant amount requested

: US $100,000

Grant budget rationale

To cover team salaries and infrastructure costs.

The requested grant amount ensures the successful development and deployment of the Obsidion Wallet over the 12-week timeline, aligning with key milestones and long-term sustainability.

1. Team Salaries

Our team consists of three experienced software engineers, specialising in blockchain, zero-knowledge, front-end, and back-end technologies.

Reasonable team salaries ensures that the team is fully dedicated to the project, covering full-time work and engagement with potential issues during the development and test launch phases.

1. Infrastructure and Development Costs

The infrastructure budget includes cloud services and testing environments needed for the development of a secure, privacy-centric wallet.