Title: AzGuard Wallet - Secure and Privacy-First Wallet for Aztec

Contact Details:

Email: [ankitagrawal620@gmail.com

](mailto:ankitagrawal620@gmail.com)Telegram: ankitagrwal

Discord: ankitagrwal

Summary:

AzGuard is a secure, privacy-centric wallet designed specifically for the Aztec network, delivering a seamless and user-friendly onboarding experience. The wallet will be browser-based with support for iFrame embedding, making it highly versatile and integrable with other decentralized applications (dApps). The project will prioritize key features like account contract management, syncing, backup, and migration functionalities, along with strong privacy mechanisms to prevent PXE data leaks. With an Alpha build 1 repo, Along with the Demo We've laid a strong foundation for future development.

We aim to move away from conventional 12-word seed phrases, using more modern methods such as passkeys to ensure security without compromising user convenience. AzGuard will also pay for the deployment of account contracts and enable first-time users to interact with private functions without immediately incurring gas fees, similar to StarkNet's account activation model. AzGuard is positioned to become a core wallet solution within the Aztec ecosystem.

Estimated Start and End Date:

- Start Date: October 2024
- End Date: Mid-Q1 2025
- Testnet Release: December 2024

About You:

We are a team of three engineers specializing in frontend, backend, and blockchain development. With prior contributions to Aztec's Sandbox and Noir contracts, we are well-versed in building privacy-preserving systems. We have 8 years of experience developing highly scalable applications and have been recognized in the Web3 space, winning the Polygon CDK Partner Prize at ETH Global.

Github: ankit875 (Ankit Agrawal) · GitHub

Details:

Design:

AzGuard will be a browser-based wallet, with the flexibility of being embedded as an iFrame in decentralized applications. The wallet will integrate with Wallet Connect, providing compatibility across various platforms. Below is a summary of the key features:

1. Account Contract:
2. The wallet will support the generation of multiple key pairs (signing key, nullifier key, incoming viewing key, outgoing viewing key) from a master seed or multiple seed phrases.
3. The account contract will be written in Aztec.nr and will implement secp256r1 for secure signing. Authwit support will be included for private and public authorizations.
4. The contract will be upgradeable, allowing for the migration of keys and contracts as necessary, supporting nonce abstraction, replay protection, and transaction cancellation.
5. Onboarding:
6. Streamlined onboarding process with minimal steps. Users will not need to manage a 12-word seed phrase; instead, passkeys will be encouraged for enhanced security.
7. Account contracts will be deployed upon the user's first transaction, avoiding the need for upfront gas fees.
8. Account Management:
9. The wallet will support syncing, backups, and migration, ensuring that users can manage multiple accounts and move between devices with ease. Snapshots will be used to sync notes and keys across devices.

10. A secure connection will be established to the PXE, preventing unauthorized access to sensitive data.
11. Privacy and Data Leaks Prevention:
12. Privacy features will prevent PXE data leaks, with a focus on user-controlled access to personal information. Data sent to the wallet's node for transaction queries will be scrubbed of any privacy-leaking content.
13. Contract Interactions:
14. AzGuard will integrate with a token bridge to allow easy bridging of tokens within the wallet. A dedicated fee payment mechanism will be implemented to enable users to pay transaction fees using their own tokens.

Flow Chart: [

1600×1280 104 KB

](https://europe1.discourse-cdn.com/flex013/uploads/aztec/original/2X/1/16b02f75667c7d9ba3c6e6b55f8e3b99bb69de53.png)

Milestone 1: Project Initialization and Technical Design (Week 1-2: October 2024)

- Objective:
- Set up the development environment, finalize technical architecture, and begin the implementation of key components. This includes team alignment, ensuring proper task allocation, and confirming that all developers have access to necessary tools and documentation.
- Set up the development environment, finalize technical architecture, and begin the implementation of key components. This includes team alignment, ensuring proper task allocation, and confirming that all developers have access to necessary tools and documentation.
- Tasks:
- Finalize wallet architecture design, including the browser-based (iframe) and WalletConnect integration.
- Develop system diagrams for the key flow: key generation, contract deployment, transaction signing, and contract interaction.
- Research and begin implementing account contract standards using Aztec.nr, including secp256r1 for signing keys.
- Finalize wallet architecture design, including the browser-based (iframe) and WalletConnect integration.
- Develop system diagrams for the key flow: key generation, contract deployment, transaction signing, and contract interaction.
- Research and begin implementing account contract standards using Aztec.nr, including secp256r1 for signing keys.
- Deliverables:
- Complete technical design documentation.
- Initial framework setup (React.js front end, backend Node Js).
- Development of wallet's UI wireframes and key design elements.
- Complete technical design documentation.
- Initial framework setup (React.js front end, backend Node Js).
- Development of wallet's UI wireframes and key design elements.

Milestone 2: Onboarding, Key Management, and Contract Registration (Week 3-5: October – November 2024)

- Objective:
- Develop and implement the user onboarding flow, key generation, and contract registration process. Ensure that the user experience is smooth, with minimal steps and privacy-preserving features integrated.
- Develop and implement the user onboarding flow, key generation, and contract registration process. Ensure that the user experience is smooth, with minimal steps and privacy-preserving features integrated.
- Tasks:

- Develop streamlined onboarding UI/UX with optional passkey support to replace traditional seed phrases.
- Implement key generation mechanisms, supporting the signing, nullifier, incoming viewing, and outgoing viewing keys.
- Build a feature to automatically generate contract addresses deterministically (without upfront deployment) based on the user’s signing public key.
- Implement functionality to delay contract deployment until the first transaction (similar to StarkNet’s account activation).

- Develop streamlined onboarding UI/UX with optional passkey support to replace traditional seed phrases.
- Implement key generation mechanisms, supporting the signing, nullifier, incoming viewing, and outgoing viewing keys.
- Build a feature to automatically generate contract addresses deterministically (without upfront deployment) based on the user’s signing public key.
- Implement functionality to delay contract deployment until the first transaction (similar to StarkNet’s account activation).

- Deliverables:
- Working onboarding flow, key generation process, and contract registration UI.
- Testable version of the wallet with key pairs generated behind the scenes.
- Delayed contract deployment implemented.

- Working onboarding flow, key generation process, and contract registration UI.
- Testable version of the wallet with key pairs generated behind the scenes.
- Delayed contract deployment implemented.

Milestone 3: Account Contract Development and Authwit Integration (Week 6-7: November 2024)

- Objective:
- Develop and integrate the account contract with core wallet functionality. Include authwit (authorization witness) support for both private and public transactions.

- Develop and integrate the account contract with core wallet functionality. Include authwit (authorization witness) support for both private and public transactions.

- Tasks:
- Write the account contract using Aztec.nr, including nonce abstraction, replay protection, and support for cancelling pending transactions.

- Write the account contract using Aztec.nr, including nonce abstraction, replay protection, and support for cancelling pending transactions.

- Deliverables:
- Fully functional account contract ready for integration with the wallet.
- Authwit support for both private and public transactions.

- Fully functional account contract ready for integration with the wallet.
- Authwit support for both private and public transactions.

Milestone 4: PXE Integration (Week 8-9: November 2024)

- Objective:
- Implement account syncing, backup features, and establish secure connections with the Private Execution Environment (PXE) to prevent data leaks.

- Implement account syncing, backup features, and establish secure connections with the Private Execution Environment (PXE) to prevent data leaks.

- Tasks:
- Establish secure communication channels with the PXE, ensuring that no unauthorized access or data leaks occur.

- Establish secure communication channels with the PXE, ensuring that no unauthorized access or data leaks occur.

- Deliverables:
- PXE integration with secure connection protocols, protecting user data.
- Wallet testnet version ready with full account management functionality.

- PXE integration with secure connection protocols, protecting user data.
- Wallet testnet version ready with full account management functionality.

Milestone 5: Contract Interactions, Fee Payment, and Final Testnet Release (Week 10-11: December 2024)

- Objective:
- Develop and integrate advanced contract interaction features, including private/public token transfers, fee payments.

- Develop and integrate advanced contract interaction features, including private/public token transfers, fee payments.

- Tasks:
- Build UI for contract interactions, including private and public token transfers.
- Implement support for batch transaction submission (app_payload to the account contract's entrypoint).
- Integrate fee payment options, including "fee juice".

- Build UI for contract interactions, including private and public token transfers.
- Implement support for batch transaction submission (app_payload to the account contract's entrypoint).
- Integrate fee payment options, including "fee juice".

- Deliverables:
- Completed wallet UI for contract interactions.
- Testnet version with all key functionalities ready for public testing.

- Completed wallet UI for contract interactions.
- Testnet version with all key functionalities ready for public testing.

Milestone 6: Post-Testnet Enhancements and Mainnet Preparation (Week 12: December 2024 – January 2025)

- Objective:
- Gather feedback from testnet users, resolve issues, and prepare for a mainnet launch. Implement post-testnet feature enhancements and optimizations.

- Gather feedback from testnet users, resolve issues, and prepare for a mainnet launch. Implement post-testnet feature enhancements and optimizations.

- Tasks:
- Collect user feedback and address any bugs or issues reported during the testnet phase.
- Improve user interface, transaction workflows, and performance optimizations based on testnet feedback.

- Collect user feedback and address any bugs or issues reported during the testnet phase.
- Improve user interface, transaction workflows, and performance optimizations based on testnet feedback.

- Deliverables:
- Final version of the wallet ready for the deployment.
- Improved and optimized post-testnet UI/UX based on feedback.

- Comprehensive documentation for users and developers.
- Final version of the wallet ready for the deployment.
- Improved and optimized post-testnet UI/UX based on feedback.
- Comprehensive documentation for users and developers.

Long-Term Roadmap (Future Plan):

- Q1 2025 (Post-Mainnet Release):
- Launch the mainnet version of AzGuard.
- Continue to add new features such as enhanced privacy controls, expanded fee payment options, and deeper integrations with Aztec ecosystem tools.
- Work towards becoming a full-featured, standalone wallet solution, offering advanced contract interactions, a plugin ecosystem, and compatibility with other L2 solutions.
- Begin work on developing a cross-chain functionality that allows AzGuard to interact with other privacy-preserving blockchains.
- Launch the mainnet version of AzGuard.
- Continue to add new features such as enhanced privacy controls, expanded fee payment options, and deeper integrations with Aztec ecosystem tools.
- Work towards becoming a full-featured, standalone wallet solution, offering advanced contract interactions, a plugin ecosystem, and compatibility with other L2 solutions.
- Begin work on developing a cross-chain functionality that allows AzGuard to interact with other privacy-preserving blockchains.

Grant Amount Requested:

- Amount: $40,000
- Frontend Engineer: $14,000
- Backend Engineer: $12,000
- Blockchain Engineer: $14,000
- Frontend Engineer: $14,000
- Backend Engineer: $12,000
- Blockchain Engineer: $14,000

Grant Budget Rationale:

The requested $34,000 covers the salaries of the three engineers over the 12-week development period, with frontend, backend, and blockchain development tasks divided proportionally. This ensures a rapid and streamlined development process that aligns with our milestones, delivering a testnet-ready wallet by December 2024. Each engineer will be responsible for their specialized area to ensure high-quality code and feature implementation across all layers of the wallet.

The budget is broken down as follows:

- Frontend Engineer ($14,000): Responsible for designing and implementing the UI/UX, integrating WalletConnect, onboarding flows, and contract interaction interfaces.
- Backend Engineer ($12,000): Focused on syncing, backups, PXE integration, and wallet infrastructure.
- Blockchain Engineer ($14,000): Account contract development, key generation, authwit integration, and contract interactions.

This allocation reflects the complexity and time required for each domain of the project, ensuring that all aspects of the wallet are developed efficiently and thoroughly.