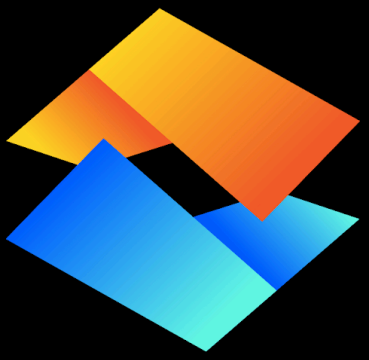

zkFold

zkFold Symbolic

a Zero-Knowledge Application Framework

zkfold.io

Content

Name of project and Project URL on IdeaScale/Fund

ZKFold Symbolic: a Zero-Knowledge Smart Contract Language
https://cardano.ideascale.com/c/cardano/idea/113453

Project Number

Project ID 1100297

Name of project manager

zkFold team

Date project started

01 April 2024

Date project completed

01 October 2024

Objectives and status of the project

Objective: The zkFold Symbolic project aimed to develop a new Haskell DSL for writing zero-knowledge smart contracts on the Cardano blockchain. The goal was to simplify the development process by abstracting the complexities of ZK cryptographic protocols, allowing developers to build efficient, scalable smart contracts without requiring expertise in ZK cryptography.

Project Status: The project was completed within the timeline and delivered all expected outputs.

List of challenge KPIs and how the project addressed them

1. **Develop and implement robust basic types and operations.**

   zkFold Symbolic is a zero-knowledge smart contract language written in Haskell. It has a mathematical primitive called arithmetic circuits as its target. To create a zero-knowledge smart contract language, we must implement a compilation algorithm for every type or function that one needs to create Cardano smart contracts. We start with more basic types and operations and proceed to more complex and Cardano-specific ones.

   Addressed by: Delivering Haskell modules for boolean, fixed-size integer, fixed-size bytestring, and fixed-size arrays. This included operations for equality, comparison, branching, and hashing (SHA2 and MiMC).

2. **Create an efficient on-chain verifier for Plonk**

   Implement the Plonk on-chain verifier. Implement an end-to-end zkFold Symbolic smart contract test on testnet.

   Addressed by: Developing Haskell modules for the Plonk on-chain verifier and testing them end-to-end using zkFold Symbolic smart contracts on the testnet.

3. **Build a library for zkFold Symbolic on Cardano.**

   Create the zkFold Symbolic Cardano Library. It will contain all necessary types to work with Cardano transactions inside zkFold Symbolic smart contracts. Implement the post-processing algorithms that convert a zkFold Symbolic script into the arithmetic circuit that is verified on-chain.

   Addressed by: Designing and implementing modules for handling Cardano transactions in zkFold Symbolic contracts and creating post-processing algorithms to integrate these scripts into an arithmetic circuit verified on-chain.

4. **Introduce variable-size types and demonstrate practical applications.**

   Implement variable-size types and operations with them (analogous to Integer and ByteString types in Haskell). Develop a set of examples, showcasing zkFold Symbolic smart contracts.

   Addressed by: Delivering modules for Integer and ByteString and showcasing their functionality through five distinct zkFold Symbolic smart contracts.

List of project KPIs and how the project addressed them

Quality of Implementation:

- Delivered well-documented, modular Haskell code for all specified tasks.
- Successfully implemented core algorithms, including SHA2, MiMC, and the Plonk verifier.

Testing and Verification:

- Verified outputs against predefined benchmarks to ensure accuracy.

Scalability and Usability:

- Built a comprehensive library (zkFold Symbolic Cardano Library) for transaction management and integration.
- Provided scalable post-processing algorithms for zkFold Symbolic scripts.

Collaboration and Open Access:

- Ensured community involvement by open-sourcing key modules and collecting feedback throughout the process.

Key achievements (in particular around collaboration and engagement)

Technical Excellence:

- Successful implementation of core functionalities, such as hashing and branching computations.

Collaboration:

- Partnered effectively with the Cardano community to refine specifications and receive iterative feedback.

Engagement:

- Engaged with developers through open-source contributions, testnet demonstrations, and documentation workshops.

Community Enablement:

- Empowered the Cardano developer ecosystem with tools and templates to create zkFold Symbolic smart contracts.

Key learnings

Technical Insights:

- The complexity of integrating zkFold Symbolic with Cardano required extensive testing and modular design.
- Post-processing for the Plonk algorithm highlighted areas for optimization in arithmetic circuit generation.

Community Feedback:

- Early and iterative engagement with developers reduced integration friction and improved library adoption.

Operational Challenges:

- Balancing scalability with the efficiency of Haskell modules demanded prioritization and innovative problem-solving.

Next steps for the product or service developed

Enhance and Expand the zkFold Symbolic Cardano Library:

- Extend functionality with additional variable-size type operations.
- Incorporate advanced cryptographic primitives to enhance security.

Optimize the Plonk Verifier:

- Focus on reducing gas costs for on-chain verification.
- Conduct further optimization to improve transaction throughput.

Develop New Use Cases:

- Partner with dApp developers to build innovative zkFold Symbolic smart contracts.
- Showcase five existing contracts as templates for real-world applications.

Community Building:

- Host workshops and hackathons to encourage adoption.
- Provide detailed documentation and example use cases.

Monitoring and Feedback:

Establish metrics to monitor library usage and effectiveness.
Implement regular updates based on user feedback and ecosystem needs.

Final thoughts/comments

The project has the potential to significantly impact the development of smart contracts on Cardano, reducing fees and increasing efficiency. zkFold's team's collaboration and careful planning were key to the project's success.

Providing developers with a proper tool to tap into the power of zero-knowledge protocols enables them to create smart contracts and DApps that are on another level in terms of efficiency and user experience. Moreover, with more developers adopting this approach, we are reducing the on-chain data footprint and compressing user transactions, effectively scaling Cardaano on layer one.

Links to other relevant project sources or documents.

A user documentation portal: https://docs.zkfold.io/symbolic/coding/getting-started/

Link to Close-out video

Close out video (short): https://youtu.be/SSHFOXrb3W4

Close out video (extended): https://youtu.be/XXNeAGuA0Is