Litepaper - Mastermind dapp: A proof of concept using ZK-Snarks scheme and Hydra protocol on Cardano.

Authors: Antonio Hernández, Juan Salvador y Agustín Salinas

Abstract

With the emergence of zero-knowledge proofs as a new field of innovation in the industry, there is an opportunity to bring these advancements to the Cardano ecosystem. However, achieving this requires the implementation of key pieces of technology tailored specifically for Cardano, this imply the technical challenge of working around the resource limitations of the network while maintaining the block propagation smooth. Our project aims to provide a solution to this technical problem through a proof of concept that implements a verifier contract for the Groth 16 protocol, leveraging the Hydra protocol for computationally demanding transactions to overcome the resource threshold. The result of this work is a dapp that adapts the Mastermind game, which serves as an ideal test case for ZK proofs.

An opportunity for Cardano in the rise of ZK technologies.

Recent innovations in the field of cryptography, specifically the surge of techniques based on zero-knowledge proofs (ZKPs), are leading to a significant evolution within the blockchain ecosystem. These advancements open up a wide range of new applications, with the most promising progress being made in the areas of privacy and scalability, which were previously insufficiently addressed or even outright neglected in the early stages of the industry. This seems to hold especially true when considering that the usual approach to resolving these deficiencies involves the aid of centralized solutions, which goes against the principle of decentralized design upon which this new internet is intended to be built. ZKPs bring innovative solutions to scalability and privacy issues, offering the possibility to address these problems directly within the protocols, thereby reducing the reliance on centralized elements that, despite their convenience, compromise the project of decentralization.

Currently, numerous initiatives have emerged in the blockchain industry to develop this new technology. For instance, we can highlight those that have arisen within the Ethereum ecosystem, where there has been a proliferation of open-source projects such as libraries, frameworks, and other tools that accelerate the design and implementation of zero-knowledge proofs in dapps. This situation is also relevant for the Cardano ecosystem, as it presents an opportunity to leverage the advancements that have emerged in other ecosystems and enable Cardano to achieve faster maturity in the field of ZK technology. This avoids the need to "reinvent the wheel" for many of the processes involved in deploying ZK applications, and it leverage the efforts already made in other areas of the industry.

The technical challenges of zero-knowledge proofs in Cardano.

However, in order for this leveraging to take place, some minimal developments are required, particularly the ability to verify ZK proofs on-chain. In this sense, Cardano is still in a very early stage, as it does not yet have a clear and definitive solution for handling ZKP verifications on the blockchain. This difficulty is partly due to Cardano being designed to ensure proper propagation and verification of blocks, which involves setting limits on the use of network resources. This condition prevents the direct implementation of ZK algorithms in smart contracts, as they require high resource consumption. Some mathematical operations of these algorithms involve processing very large finite number fields, exceeding the computational budget of the network. Our solution to this technical limitation was to delegate computationally intensive transactions to sidechains, particularly the Hydra protocol. This decision is driven by several relevant reasons related to this problem and other similar ones, which are important to point out:

1. Avoiding overloading the network requires understanding that computationally intensive computations must be executed by all nodes in the network within a limited time frame.
2. Not having resource constraints that hinder the development of smart contracts with complex functionalities.

3. Maintaining transactions at a low cost.
4. Due to the isomorphic property of Hydra, when network resources eventually increase, the implemented contracts could be redeployed directly on the mainnet.

Next, we will explain the proof of concept we conducted in order to address this challenge within our ecosystem and pave the way for the development of ZK applications.

Proof of Concept: A Dapp with ZK-Verifiable Contracts on Hydra.

As part of our proof of concept, we developed a Dapp based on the game Mastermind. Mastermind is a two-player game where one party provides hints about a secret code, and the other player tries to guess the code to win. This game dynamics proved ideal for applying zero-knowledge proofs because, in each round, the hints need to be demonstrated without being revealed. To achieve this, we implemented the Groth 16 protocol on the Cardano network. This protocol utilizes a ZK-Snark proof, which is verified on-chain through a smart contract. Initially, we attempted to generate transactions on the mainnet but encountered limitations due to the resource constraints mentioned previously. This justified the subsequent use of the Hydra protocol to process computationally intensive transactions. This milestone was a key breakthrough in our project and, in our opinion, opens up promising possibilities for the development of complex Dapps.

The Mastermind Game

Originally, Mastermind is a two-player board game that involves guessing a secret combination of colors. On one side, there is the "code maker" who begins the game by generating a secret combination, which the "code breaker" must guess within a certain number of attempts to win. In each incorrect attempt by the "code breaker," the "code maker" provides a clue: a black peg for each element of the combination that is the correct color and in the correct position, and a white peg for each element that is the correct color but in the wrong position.

For example, if the secret combination is RRVA (red-red-green-blue) and the "code breaker" tries to guess with "MRVR" (purple-red-green-red), the clue given by the "code maker" would be:

- Two black pegs (indicating that the red and green pegs are both the correct color and in the correct positions).
- One white peg (indicating that the final red peg is the correct color but in the wrong position).
- No pegs for the purple peg, as it is neither the correct color nor in the correct position.

Lately, in our adaptation as a Dapp, the "code breaker" wins if they guess the secret combination within a certain number of attempts. It has become a game of gambling, where if the "code breaker" wins, the smart contract allows them to claim the Ada (cryptocurrency), and if they lose or abandon the game after a certain time, the "code maker" can claim the Ada. It is worth noting that in the game design, both the number of colors to guess and the number of attempts before losing can be arbitrary. The crucial factor for fair gameplay is that the clues should not include any traps or misleading information. This is where the technology of zero-knowledge proofs comes into play. It ensures that the clues provided by the "code maker" are honest without revealing the secret combination. It is essential to explain the significance of this in order to understand the functionality and scope of the project. By utilizing zero-knowledge proofs, the integrity of the game is upheld, allowing for a fair and secure gaming experience where the secrecy of the secret combination is maintained.

Zero-knowledge proofs and its applications

Zero-knowledge proofs (ZKPs) are cryptographic demonstrations that allow one party to prove that a secret parameter or witness satisfies a certain truth condition without revealing it to a verifying counterpart. Based on this proof, we can be certain that if the prover does not know the secret parameter, they cannot construct a valid proof. However, if they possess the knowledge, they can send the proof instead of the actual secret.In the

case of the Mastermind game, the "code maker" provides a proof that the clue they give is correct without revealing it. Essentially, it demonstrates a certain state of affairs without exposing the content that is being proved. This is highly counterintuitive to our usual way of proving things in everyday life, where we simply show what we believe to know, and a third party makes a judgment.In this regard, zero-knowledge proofs have opened up numerous use cases, but in the context of the blockchain industry, at least two relevant applications stand out:

- **Privacy**: Zero-knowledge proofs allow for the verification of certain properties without revealing the underlying data. This is particularly useful in privacy-focused applications, such as anonymous transactions or identity verification, where sensitive information needs to be protected while still proving the validity of the transaction.This condition can be expressed as "It is true or false that the prover knows a certain piece of information."
- **Scalability**: Zero-knowledge proofs can address the scalability challenges faced by blockchain networks. By offloading computationally intensive operations to the off-chain layer, such as proof generation, and verifying the proofs on-chain, the burden on the main blockchain is reduced, allowing for more efficient and scalable transactions.. These verify if certain transactions have occurred according to the rules of the protocol, thereby ensuring that certain transactions are taking place in a distributed ledger without directly computing and storing those transactions. The truth condition being proven is "Did a valid transaction occur within the chain or not?"

Indeed, within the domain of ZK technology, a type of cryptographic proof called ZK-Snarks (Zero-Knowledge Succinct Non-Interactive Argument of Knowledge) has gained prominence. Due to their compact size, which is ideal for the storage requirements of blockchain networks, as well as their ease and speed of verification, requiring only a single transaction (as they are non-interactive proofs), these schemes have become prominent in the industry. These characteristics make them valuable for the performance of a blockchain network, where efficiency and scalability are crucial. This is important for achieving true decentralization, as proofs with higher resource requirements would limit

participation to individuals with significant resources, jeopardizing the heterogeneity and decentralization inherent in permissionless protocols. For our project, we utilized the Groth 16 protocol, which follows a ZK-Snark scheme.

A three-phase demonstration

A zk-Snark scheme consists of three algorithms:

1. Setup: $S(C, \lambda) = (pk, vk)$

In this process, two public keys are generated, a proving key pk and a verification key vk. To generate these keys, an arithmetic circuit (representing the algorithm to be proven) and a random secret parameter λ (lambda) are required. λ is crucial for ensuring the authenticity and cryptographic security of the proof. If λ is compromised, false proofs could be generated. In the case of the proof of concept, the circuit C represents the Mastermind game as a series of arithmetic operations. This representation is important as it allows for the testing of true or false computational statements within the context of the game, where zero represents true and any non-zero number represents false.

2. Proof: $P(pk, x, w) = \pi$

The proof generation function P takes as input the proving key (pk), a public argument (x), and a witness (w), which is the secret being proven. It generates a proof (π) that demonstrates that the prover knows the witness (w) and that the witness satisfies a truth condition within the program's context. In the case of the game, the "codifier" generates a proof for a clue (π) using a proving key (pk), a clue as the public value (x), and the secret combination of the guess (w).

3. **Verification**: $V(vk, x, \pi)$ = True or False

The verification function V, using the verification key (vk), public parameter (x), and the proof (π), determines whether the proof is valid or not. This function aims to provide certainty that a prover knows a valid witness (w) along with a valid public parameter (x)

in relation to an arithmetic circuit C (the representation of the game). The function V will return true or false depending on whether the prover can generate a proof that satisfies the aforementioned criteria.

The Mastermind dapp

Regarding the ZK-Snark scheme, we followed a relatively common design pattern in the industry to develop the dapp. The off-chain processes involve the setup and generation of the proof, while the verification of the proof takes place on-chain. We utilized circom to generate the circuits and snarks to generate the proofs. These steps are computed in the browser simultaneously with the players' interactions. On the other hand, the proofs are verified in contracts written in the Plutus language, which also manage the game logic. The main idea behind this development pattern is to prioritize resources and ensure that the verification process has the security guarantees inherited from using a blockchain network. In general terms, the dapp functions as follows:

1. **Fund Collection:** The first player, who takes on the role of the "encoder," will send funds to be collected in a Hydra smart contract. Then, the second player, who takes on the role of the "decoder," will also send funds as collateral to the same contract to enter the game. Once the funds from both players are collected, the Hydra protocol can be initiated, and the subsequent transactions will occur on the sidechain.
2. **Game Start:** At this point, the "encoder" can start the game by sending a hash of the secret combination.
3. **Gameplay:** The game unfolds as a series of turns where the "decoder" guesses the secret combination, and the "encoder" responds with hints. Each hint will be accompanied by a proof that is verified using a ZK-Snark scheme, ensuring that the "encoder" cannot provide dishonest hints to the opponent.
4. **End of Game and Distribution:** If the "decoder" manages to guess the secret combination within a certain number of attempts, the game will end. At that point, the Hydra protocol can be closed, and the funds will be distributed according to the winner on the main network.

Possible applications of this proof of concept.

Next, we will highlight some of the applications that necessarily require the existence of ZK-Snark scheme validators in smart contracts. These applications emphasize the importance of this project and the possibilities that arise from its success:

1. **Bridges and cross-chain protocols:** Currently, ZK technology is being used to provide greater security guarantees for interactions between chains, particularly through light clients to verify that certain events and transactions are occurring on a chain. This is a significant improvement over the previously seen commissions and multi-key models that have been the target of computer attacks in recent times. Some examples in this area include zkBridge, zkSync, zkEvmos, etc.
2. **Roll-ups**: These are transactions that are compressed into a proof and then verified on the main chain, bringing improvements in terms of scalability, both in cost and efficiency.
3. **Anonymous Swaps**: Another application case is anonymous and confidential token exchanges, where ZKPs are used to hide both the balances and the recipients of the transactions.

While there are more applications for this technology, these are, in our opinion, the ones that would have the greatest impact on the Cardano ecosystem in the context of Layer 1, advancing areas of the network such as scalability, interoperability, and privacy.

Projections as a team and entry into the market.

As a team, our mission is to leverage our technology expertise to serve emerging projects on Cardano. We aim to be catalysts for innovation in the industry. In this regard, our work and services are directly targeted towards web3 companies and developer communities within the ecosystem. Specifically, there are some key points that encompass both our overall strategy and the one we project for this particular project:

1. Generate new products and services that make the use of these innovative technologies, such as zero-knowledge proofs, simpler and more accessible. Currently, this area has generated interest within the ecosystem, especially among DeFi projects.
2. Form alliances with other actors to collaborate, create synergy, and promote awareness of our services and products.
3. Create educational content about cryptography, zero-knowledge proofs, and blockchain development.
4. Contribute to and maintain an active presence in the Cardano communities, with a particular focus on the Spanish-speaking regions to which we belong.

Conclusion: Future Challenges

It is essential to emphasize the importance of this task for the Cardano ecosystem, as a mature solution could catalyze other significant innovations in the ecosystem. Additionally, we acknowledge that there are other relevant solutions to this problem, such as the addition of primitive functions to the protocol that partially or fully execute ZK algorithms (as being worked on in CIP-85 and CIP-381). This would have the benefit of making transaction execution more efficient and preserving the network's security properties. Ultimately, we hope that both our contributions and the advancements made in this area, including the aforementioned ones, will allow for the execution of ZK algorithms at Layer 1 in the near future. This will maximize the potential of this technology and ensure that Cardano is competitive with other protocols like Ethereum. We also want to highlight our experience working with the Hydra protocol. We hope that the advancements resulting from this project will enhance Hydra's fit-to-market as a product by expanding its use cases.

Resources

1. “What is a ZK-Snark ", Blockhain-Web3 MOOC channel. Youtube.
2. “Zero Knowledge Proofs”, Computerphile channel. Youtube.
3. Hackathon 101: An introduction to Zero-Knowledge Proofs. Emurgo blog.
4. Introduction to zk-Snarks with examples, Christian Lundkvist. Medium.
5. Mastermind board game, Wikipedia.
6. “Zero-knowledge proofs, a board game, and leaky abstractions: how I learned to write zk-SNARKs”, Koh Wei Jie. Medium.
7. ZK Tech You Should Know — Part 1: SNARKs & STARKs, Mina protocol channel. Youtube.
8. ZK Tech You Should Know — Part 2: zkRollups, Mina protocol channel. Youtube.
9. zkBridge: Trustless Cross-chain Bridges Made Practical Tiancheng Xie , Jiaheng Zhang , Zerui Cheng , Fan Zhang , Yupeng Zhang , Yongzheng Jia , Dan Boneh , Dawn Song.