Slovak University of Technology in Bratislava
Faculty of informatics and information technologies

Lukáš Častven

Designing Zero-Knowledge Proof Solutions in Ethereum ecosystem

Progress report on DP1 solution

Thesis supervisor: Ing. Kristián Košťál PhD.

May 2025

Slovak University of Technology in Bratislava

Faculty of informatics and information technologies

Lukáš Častven

Designing Zero-Knowledge Proof Solutions in Ethereum ecosystem

Progress report on DP1 solution

Study programme: Intelligent software systems

Study field: Computer Science

Training workplace: Institute of Computer Engineering and Applied Informatics

Thesis supervisor: Ing. Kristián Košťál PhD.

May 2025

SLOVENSKÁ TECHNICKÁ
UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY
A INFORMAČNÝCH TECHNOLÓGIÍ

Návrh zadania diplomovej práce

Finálna verzia do diplomovej práce [1]

Študent:

Meno, priezvisko, tituly:	Lukáš Častven, Bc.
Študijný program:	Inteligentné softvérové systémy
Kontakt:	xcastven@stuba.sk

Výskumník:

Meno, priezvisko, tituly:	Kristián Košťál, doc. Ing. PhD.

Projekt:

Názov:	Návrh riešení využívajúcich dôkazy s nulovým vedomím v Ethereum ekosystéme
Názov v angličtine:	Designing Zero-Knowledge Proof Solutions in Ethereum ecosystem
Miesto vypracovania:	Ústav počítačového inžinierstva a aplikovanej informatiky, FIIT STU
Oblasť problematiky:	blockchain, kryptografia, kryptomeny, dôkazy s nulovým vedomím

Text návrhu zadania[2]

Zero-knowledge proofs (ZKPs) are a new cryptographic primitive in applied cryptography with applications in multiple industries, including Web3, supply chains, and the Internet of Things. By verifying the authenticity of information without disclosing its content, ZKPs improve privacy, security, and efficiency in digital systems. Current use cases include decentralized identity (Worldcoin), private transactions (stealth address schemes or blockchains like Zcash and Monero), secure and scalable Layer-2s (ZkSync, Scroll) voting systems, IoT networks, and supply chain management.

Examine existing solutions, proposals, and trends in this domain. Analyse a specific challenge discovered through the related work. Design a solution to address the challenge. Implement and test the solution on Ethereum (or a Layer-2) blockchain network. Evaluate and compare results with existing approaches. Discuss findings and contributions. Conclude with novelty, scientific findings, and future research directions.

[1] Vytlačiť obojstranne na jeden list papiera

[2] 150-200 slov (1200-1700 znakov), ktoré opisujú výskumný problém v kontexte súčasného stavu vrátane motivácie a smerov riešenia

Literatúra[3]

- Ulrich Haböck, David Levit, Shahar Papini. Circle STARKs. Cryptology ePrint Archive, 2024, https://eprint.iacr.org/2024/278.
- Jeremy Bruestle, Paul Gafni, and the RISC Zero Team. RISC Zero zkVM: Scalable, Transparent Arguments of RISC-V Integrity. Risc0. Retrieved January 11, 2024 from https://dev.risczero.com/proof-system-in-detail.pdf.

Vyššie je uvedený návrh diplomového projektu, ktorý vypracoval(a) Bc. Lukáš Častven, konzultoval(a) a osvojil(a) si ho doc. Ing. Kristián Košťál, PhD. a súhlasí, že bude takýto projekt viesť v prípade, že bude pridelený tomuto študentovi.

V Bratislave dňa 1.6.2025

Podpis študenta	Podpis výskumníka

Vyjadrenie garanta predmetov Diplomový projekt I, II, III

Návrh zadania schválený: áno / nie[4]

Dňa: .....

Podpis garanta predmetov

[3] 2 vedecké zdroje, každý v samostatnej rubrike a s údajmi zodpovedajúcimi bibliografickým odkazom podľa normy STN ISO 690, ktoré sa viažu k téme zadania a preukazujú výskumnú povahu problému a jeho aktuálnosť (uveďte všetky potrebné údaje na identifikáciu zdroja, pričom uprednostnite vedecké príspevky v časopisoch a medzinárodných konferenciách)

[4] Nehodiace sa prečiarknite

Čestné prehlásenie

Čestne vyhlasujem, že som túto prácu vypracoval(a) samostatne, na základe konzultácií a s použitím uvedenej literatúry.

V Bratislave, 31.5.2025 .....

Lukáš Častven

Anotácia

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Študijný program: Inteligentné softvérové systémy

Autor: Lukáš Častven
Priebežná správa o riešení DP1: Návrh riešení využívajúcich dôkazy s nulovým vedomím v Ethereum ekosystéme
Vedúci bakalárskej práce: Ing. Kristián Košťál PhD.
Máj 2025

Táto práca skúma návrh a potenciálnu implementáciu riešení využívajúcich dôkazy s nulovým vedomím (ZKP) v rámci ekosystému Ethereum, s cieľom riešiť problémy škálovateľnosti a dosiahnuť post-kvantovú bezpečnosť. Dôkazy s nulovým vedomím sú kryptografické metódy, ktoré umožňujú overenie integrity výpočtov bez odhalenia citlivých dát, čo ich robí mimoriadne vhodnými pre zlepšenie súkromia a efektivity v blockchainových systémoch. Ústredným zameraním tejto práce je prieskum Zero-Knowledge Virtual Machine (ZkVM) prispôsobenej pre inštrukčnú sadu RISC-V, s využitím princípov kryptografie založenej na mriežkach. Kľúčovou aplikáciou takéhoto ZkVM by bolo dokázateľne správne vykonanie Ethereum blokov cez ZkEVM.

Výskum zahŕňa analýzu architektúry Ethereum, konceptu "snarkifikácie" pre konsenzuálnu aj exekučnú vrstvu, evolúciu a prirodzené výzvy ZkEVM (vrátane rekurzívnych dôkazov a skladacích schém), a preskúmanie kryptografických primitív založených na mriežkach, ako sú Ajtaiho záväzky, spolu so súčasnými systémami LaBRADOR, Greyhound a LatticeFold/LatticeFold+.

Cieľom tejto diplomovej práce je prispieť do odboru zhodnotením poten-

ciálnej výkonnosti ZKP systémov postavených na kryptografii založenej na mriežkach pre úlohu dokazovania správneho vykonania Ethereum blokov. Práca sa ďalej snaží porovnať tieto nové riešenia s etablovanými klasickými ZKP prístupmi, pričom zachováva post-kvantovú odolnosť.

Annotation

Slovak University of Technology in Bratislava
Faculty of informatics and information technologies
Degree Course: Intelligent software systems

Author: Lukáš Častven
Progress report on DP1 solution: Designing Zero-Knowledge Proof Solutions in Ethereum ecosystem
Supervisor: Ing. Kristián Košťál PhD.
May 2025

This thesis explores the design and potential implementation of solutions utilizing Zero-Knowledge Proofs (ZKPs) within the Ethereum ecosystem, aiming to address scalability challenges and achieve post-quantum security. ZKPs are cryptographic methods that allow the verification of computational integrity without revealing sensitive data, making them suitable for improving privacy and efficiency in blockchain systems. The central focus of this work is the exploration of a Zero-Knowledge Virtual Machine (ZkVM) for the RISC-V instruction set, leveraging lattice-based cryptography. A key application of such a ZkVM would be to enable creating proofs of correct execution of EVM.

The research includes an analysis of Ethereum's architecture, the concept of "snarkification" for both consensus and execution layers, the evolution and challenges of ZkEVMs (recursive proofs and folding schemes), and an examination of lattice-based cryptographic primitives, such as Ajtai commitments, along with contemporary systems like LaBRADOR, Greyhound, and LatticeFold/LatticeFold+.

This engineer’s thesis aims to contribute to the field by evaluating the potential performance of ZKP systems built on lattice-based cryptography for the task of proving the correct execution of Ethereum blocks. The work further seeks to compare these emerging solutions with established classical ZKP approaches, while maintaining post-quantum resistance.

Table of contents

List of Figures

List of abbreviations used

EVM	Ethereum Virtual Machine
ISA	Instruction Set Architecture
L1	Layer 1
L2	Layer 2
PCS	Polynomial Commitment Scheme
PoS	Proof of stake
p2p	peer-to-peer
SNARK	Succinct Non-interactive Argument of Knowledge
STARK	Scalable Transparent ARgument of Knowledge
R1CS	Rank-1 Constraint System
ZK	Zero Knowledge
ZkEVM	Zero Knowledge Ethereum Virtual Machine
ZkVM	Zero Knowledge Virtual Machine
ZKP	Zero Knowledge Proof

Chapter 1

Introduction

Ethereum, a decentralized blockchain platform launched in 2015, provides a Turing-complete execution environment for smart contracts and decentralized applications. Its modular architecture comprises a proof-of-stake consensus layer (Gasper) and an execution layer governed by the Ethereum Virtual Machine (EVM). A primary challenge for Ethereum is scalability, as the need for all nodes to validate and re-execute every transaction limits its throughput and can centralize the network by increasing resource demands.

"Snarkification," the integration of Zero-Knowledge Proofs (ZKPs) like SNARKs, offers a path to mitigate these scalability issues. Initiatives such as the 'Beam Chain' aim to snarkify consensus, allowing validators to verify compact proofs instead of entire state components. Similarly, Zero-Knowledge Ethereum Virtual Machines (ZkEVMs) seek to make EVM execution provable, enabling nodes to verify block execution via a single proof. Enshrining a ZkEVM into Ethereum's Layer 1 could further enhance scalability and simplify the creation of native rollups.

Zero-Knowledge proofs allow proving a statement's truth without revealing underlying information. The statement are usually a membership in NP language, thus an entire computations can be proven with this primitives, for example like computations done by the EVM, turning it into a ZkEVM. However, developing ZkEVMs is challenging due to the EVM's original design: complex opcodes, large word sizes requiring range proofs, its stack-based nature, and ZK-unfriendly storage mechanisms. Early solutions utilized recursive proofs to aggregate computation steps, leading to the first generation of Layer 2 ZkEVMs. Folding schemes further optimized this by compressing instances. The current trend involves Zero-Knowledge Virtual Machines (ZkVMs), especially those based on the simpler, register-based RISC-V ISA, which is more amenable to ZK proof generation.

While many ZKP systems rely on assumptions vulnerable to quantum attacks, lattice-based cryptography provides a foundation for post-quantum secure ZK solutions, with security often based on hard problems like SVP or SIS.

This thesis investigates the design and implementation of a RISC-V ZkVM using lattice-based cryptography. A key application is proving Ethereum block execution, contributing to the network's scalability and post-quantum security. This work explores the practical construction of such a system and aims to assess whether lattice-based cryptography can offer performance comparable to classical ZKP methods while ensuring quantum resistance.

Document Structure

This progress report on the DP1 solution details the initial analytical work in Analysis 2. This chapter begins with an overview of Ethereum 2.1, covering

its consensus and execution mechanisms. It then explores the concept of Snarkification of Ethereum 2.2, discussing Zero-Knowledge Proofs and their application to both the consensus layer via the Beam Chain initiative and the execution layer through enshrined ZkEVMs. Following this, the report analyzes ZkEVMs 2.3, examining the challenges of proving EVM execution, the evolution of solutions from recursive proofs and folding schemes to modern ZkVMs. Finally, the analysis introduces Lattice-based cryptography 2.4, outlining Ajtai commitments and the potential of lattices for post-quantum zero-knowledge systems, including recent developments like LaBRADOR, Greyhound, and LatticeFold/LatticeFold+.

Chapter 2

Analysis

2.1 Ethereum

Ethereum is a decentralized, open-source blockchain with smart contract functionality [1]. Conceptualized by Vitalik Buterin in 2013/2014, it went live in 2015, aiming to build a platform for decentralized applications that adds Turing complete execution environment to blockchain [2]. Ethereum enables developers to create and deploy decentralized applications and crypto assets. The Ethereum network is comprised of two layers, two peer-to-peer networks, consensus layer and execution layer.

2.1.1 Consensus

Consensus on Ethereum is achieved with proof-of-stake (PoS) consensus mechanism called Gasper [3]. PoS system relies on crypto-economic incentives, rewarding honest stakers (people who put economic capital in the network) and penalizing malicious ones.

Stakers, or also called validators, propose new blocks. Validator is selected for a block proposal pseudo-randomly from the pool in each slot. A slot is some time amount (as of writing of this work it is 12 seconds on Ethereum mainnet), in which the pseudo-randomly chosen validator can propose new block. The software creating the new block is called consensus client. Proposer's consensus client requests a bundle of transactions from the execution layer 2.1.2, wraps them into a block and gossips (sends) the new block to other participants over the consensus p2p network. The rest of the validator pool can in 32 slot (or one epoch) attest to that new block's validity. In order for a block to be finalized, it must be attested by a super-majority, which is 66% of the total balance of all validators. [4].

2.1.2 Execution

Software operating on the execution layer is called execution client. Nodes on execution layer hold the latest state and database of all Ethereum data. These clients gossip incoming transactions over the execution layer p2p network, and each stores them in a local mempool. Once they are put inside a block, each transaction is executed in Ethereum-Virtual-Machine (EVM). EVM is a stack based virtual machine operating 256 bit words which executes smart contract code. Ethereum is a decentralized state machine, with rules defined by the EVM. EVM can be thought of as a function. Given a valid state and valid set of transactions[1], outputs next valid state:

$$F(S_n, T) = S_{n+1}$$

[1] validity of transactions, and thus transitively validity of state, is guaranteed by the consensus layer 2.1.1

Thus, Ethereum's state transition function is described by the EVM. This function must be executed by each execution node for each new block in order to keep up with the current chain state. [5]

2.2 Snarkification of Ethereum

As discussed in sections on consensus (2.1.1) and execution layer (2.1.2), nodes on each layer must expend compute resources to validate the consensus data of new blocks or re-execute all transactions within them to verify and maintain the blockchain's state. These computational demands limit the scalability of the entire network.

Ethereum's decentralized node network includes not only participants with capable servers, but also hobbyists, who use home internet connections and less powerful machines. Naively attempting to scale the network, for instance, by decreasing slot time, or increasing block size may raise bandwidth and node requirements, thereby ousting less capable nodes and thus hindering Ethereum's decentralization.

Snarkification refers to the process of integrating specific type of Zero Knowledge Proofs (ZKPs) called SNARK (Succinct Non-interactive Argument of Knowledge) in order to offload a computation to one entity (one node, or a cluster that is a small subset of the whole network), which performs the computation and generates a proof of its validity. This proof can then be verified by the rest of the network at fractional compute cost.

2.2.1 Zero Knowledge Proofs

Zero-Knowledge Proofs are cryptographic primitives that allow a prover to convince a verifier of a statement's truth without revealing any information beyond the statement's validity itself. The statement typically concerns membership in an NP language/relation [6]. The membership statements are usually named ZK circuits. Traditionally, ZKPs are interactive protocols where a prover, knows a private witness for a statement (e.g., a solution to a problem), aims to convince a verifier of this knowledge. Through a series of back-and-forth interactions, the verifier becomes convinced of the prover's claim (if true) without learning the witness itself.

Interactive proofs where the verifier's messages consist only of random challenges are known as public-coin protocols. The Fiat-Shamir heuristic [7] can transform such public-coin interactive proofs into non-interactive proofs by replacing the verifier's random challenges with challenges derived from a cryptographic hash function applied to the protocol's transcript. This non-interactivity allows a single generated proof to be validated by any number of verifiers.

A SNARK is a specific type of ZKP. The 'Succinct' aspect implies that proof sizes are very small (e.g., polylogarithmic in the size of the witness or statement) and verification is faster than re-executing the original computation, often polylogarithmic in the computation's complexity [8].

2.2.2 Snarkifying consensus

The 'Beam Chain' is a research initiative, introduced by Justin Drake at Devcon 2024 [9], aimed at redesigning and improving Ethereum's consensus mechanism. A key aspect of this proposal involves leveraging SNARKs to

make consensus layer state transitions provable, in near real-time. Instead of each validator independently re-validating all components of a new consensus state, they would verify a single, compact SNARK. Based on this proof's validity, they would accept or reject the proposed state update.[2]

2.2.3 Snarkifying execution

Ethereum's execution state transitions are dictated by the rules of the EVM. To reduce the computational burden of verifying execution using ZKPs, the EVM's execution logic needs to be provable with a ZKPs, creating a ZkEVM. This would change the state transition function (2.1.2) to:

$$F(S_n, T) = (S_{n+1}, \pi)$$

Where π is the proof of the state transition's validity. Therefore, verifying a block's execution state transition would involve verifying this single proof, rather than re-executing all transactions within the block. Furthermore, an increase in block size would have a little to no impact on verification times for nodes, as proof verification complexity typically grows much slower than the computation being proven. This could allow for larger block sizes without risking the exclusion of nodes with modest computational capabilities due to their weaker computational capabilities, thereby supporting scalability.

Another approach is to 'enshrine' a ZkEVM directly into the Ethereum protocol (L1). This offers similar benefits in terms of L1 state transition verification, as the native EVM could delegate execution proving to this enshrined ZkEVM. In addition, an enshrined ZkEVM could support and simplify the

[2]The entity proposing the state update, such as a block proposer, would be responsible for generating and submitting this proof.

creation of what are sometimes termed 'native rollups' [10]. These would be Layer 2 solutions that could leverage the L1's enshrined ZkEVM for verifying their own state transitions, which are themselves batches of EVM-compatible transactions. The correctness of these L2 transaction batches would thus be directly enforced by the L1 execution layer through ZK proof verification. This could lead to faster, settlement for these L2s on L1, which in turn could simplify composability between L1 and these native L2s, as well as among different native L2s.

2.3 ZkEVM

ZkEVM, as mentioned in 2.2.3, is a modified EVM, which given state and list of transactions produces next state and a proof π attesting to correctness of that execution. Primary use case of ZkEVMs today is enabling "ZK rollups", L2s which execute a bundle of transactions offchain, generate a proof and submit the bundle with the proof on the L1, where only the proof is validated. Second use case, but not used today, is to enshrine ZkEVM into EVM as described in 2.2.3.

2.3.1 Challenges of EVM

EVM was not designed with ZK taken into consideration:

1. Many EVM opcodes are complex and expensive to prove with ZK. This has led to different types of EVM compatibility among ZkEVMs [11].
2. 256 bit word size means that ZK systems working over prime fields must include range proofs, which increases the ZkEVM complexity.
3. It's harder to prove stack based VM. For example, Starkware developed

Cairo [12], a register based model in order to implement its ZkEVM. This requires a custom smart contract compiler.

4. EVM storage uses Merkle Patricia Tries with Keccak, which are not ZK friendly and have huge proving overhead.

Proving the entire EVM within a single circuit (one proof) is computationally and economically unfeasible. Such a proof would be several megabytes in size, making it too expensive to store on L1, let alone prove.

2.3.2 Recursive proofs

Recursive proofs combine benefits of ZKP systems with fast prover times (e.g., those based FRI) and systems with short proofs (like Groth16 [13]). The idea is to produce a proof of a knowledge of a proof.

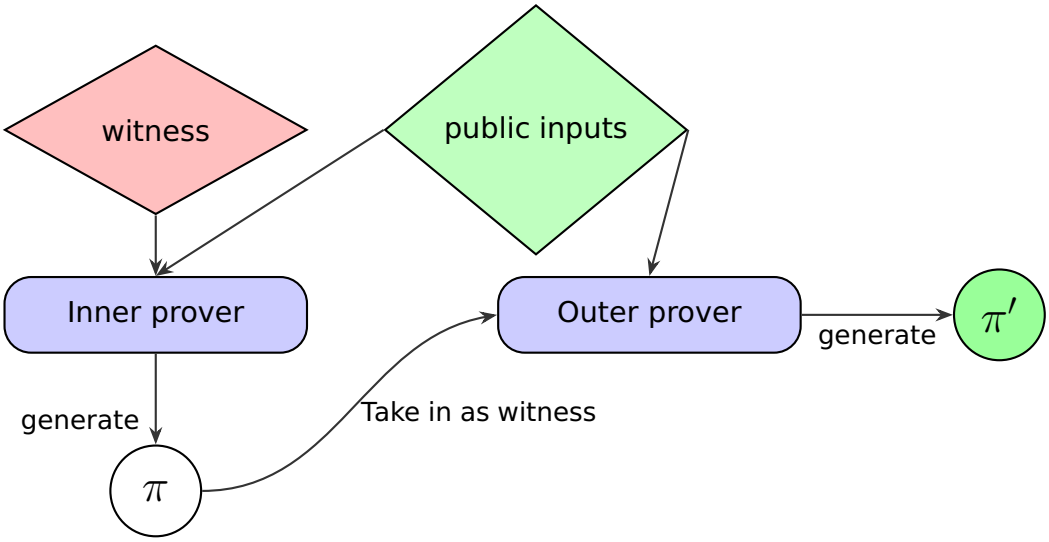

Figure 2.1: Recursive proving

The recursive ZKP is an instantiation of Incrementally-Verifiable-Computation

(IVC) [14]. IVC is a cryptographic primitive for proving the correctness of an iterated and incremental computation (such as EVM, or RISC-V microprocessor). The output of step i of the computation (F) is fed as input into step $i+1$ of the computation, along with a proof π, which proves that:

1. $F(s_{i-1}, \omega_i) = s_i$ - proves that executing F with s_{i-1} and ω_i correctly outputs s_i,
2. $V((i-1, s_0, s_{i-1}), \pi_{i-1}) = true$ - proves that π_{i-1} is a valid proof relative to the previous step of the computation.

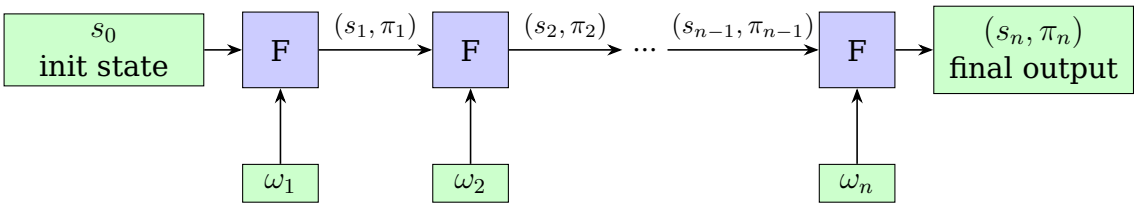

Figure 2.2: IVC [14]

Final proof π_n is a succinct proof that prover has $\omega_1, \ldots, \omega_n$, s.t., final output s_n is correct. The first implementation of IVC was done using SNARKs [15]

2.3.3 First generation of ZkEVMs

In the 2022 to early 2024 first generation of ZkEVMs was successfully deployed. Thanks to recursive proving, it was feasible for these ZkEVMs to be created. They first prove the EVM execution, and then prove it inside another circuit. This enabled parallelization of proving and reduced the size and complexity of the final proof submitted to L1. ZK systems like Plonky2 [16], or Halo2 [17], are examples of recursive ZKP systems. Notable refer-

ences include Scroll, zkSync Era, Polygon ZkEVM.

2.3.4 Folding schemes

Even though recursion proofs enabled teams to build ZkEVMs, they are not without their flaws. Prover needs to have circuit which contains the whole verification algorithm of another proof system. It must verify expensive evaluation proofs for polynomial commitments.

Folding scheme takes almost all of the verification steps out of the circuit. There are no FFTs, only multi-exponentiations, which do not require big memory overhead. Also no embedded elliptic curve pairings are needed, because there is no need to switch curves like in recursive proofs.

It compresses two instances into one. Folding prover will fold two instances, with corresponding witnesses, and produce single instance, s.t., if it is correct, it is implied that also the two original instances were correct [18].

Given some relations $R = \alpha(pp,\ x,\ W)$, where:

- pp = public parameters,
- x = instance or public inputs,
- W = witness.

For example: R is an equation, x are coefficients and W are concrete values that satisfy the equation.

A folding scheme for two relations R and R_{acc} (or R_1 and R_2) is an interactive protocol between folding prover P and folding verifier V where:

1. P has $(pp, x_1, w_1) \in R, (pp, x_2, w_2) \in R_{acc}$,

2. V has $(pp, x_1), (pp, x_2)$.

And the result of their interaction is (pp, x_3, w_3) (where V only knows (pp, x_3) and w_3 is kept private for P). Thus instead of validating two instances, verifier only needs to check one. To make the interaction between P and V non-interactive with [7], P must also provide proof that Fiat-Shamir was used correctly. And also, P needs to prove that the folded instances are all linked together, output of step i is input of step $i + 1$.

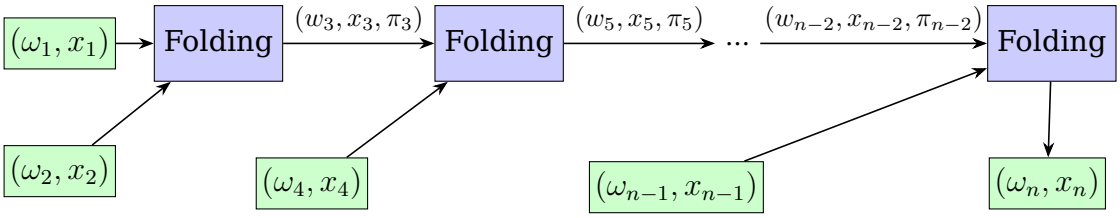

Figure 2.3: Folding scheme [18]

2.3.5 New generation of ZkEVMs

Nowadays, industry is moving towards ZkVMs, dominantly RISC-V ZkVMs. RISC-V is an open sourced Instruction Set Architecture (ISA) [19]. RISC-V's simpler, reduced instruction set is better suited for ZK proof computations. Its register-based architecture is generally more efficient for proving than the EVM's stack-based architecture. Another benefit is a mature tooling ecosystem, including compilers like GCC and LLVM, debuggers, and libraries. This allows developers to write provable programs in familiar languages like Rust, C, and C++. And RISC-V is a general purpose ISA, which opens possibilities for proving not just blockchain applications. Due to this, a EVM written in a language that can be compiled into RISC-V, can be executed inside a ZkVM and together this combination creates a ZkEVM.

RISC-0

First such ZkVM was RISC-0 [20], a general purpose RISC-V VM. It implements the RISC-V RV32IM specification (the RV32I base with the multiplication extension). It uses STARKs [21] with FRI [22], for the inner provers, and the final output proof is generated with Groth16 [13]. Proving process of RISC-0 is as follows:

1. From execution of a program a collection of segments is collected,
2. Each segment is proven with STARK based circuit [23],
3. Pairs of these proofs are recursively proven until only one proof remains,
4. This proof is proven with Groth16.

SP1

SP1 by Succinct is also a RISC-V RV32IM, proving programs in a recursive STARK [21] environment using FRI [22], with final proof being wrapped by Groth16 [13], or PlonK [24] SNARK for small proof sizes [25].

Jolt

Another RISC-V RV32I ZkVM. Uses a technique termed as *lookup singularity*. This technique produces circuits that perform only lookup into massive precomputed lookup tables of size more than 2^{128} [26]. Underlying lookup argument protocol is Lasso [27].

OpenVM

OpenVM introduces a RISC-V ZkVM with novel "no-CPU" design. This decouples opcode implementation and let's developers build custom VM extensions. Using FRI [22] and DEEP-ALI [28]. The overall proving process follows previous designs [20], but enables aforementioned modularization [29].

2.3.6 Comparison

The only sensible comparison method for ZkEVMs is to benchmark them on proving Ethereum blocks. However, this is not doable in this analysis because of two reasons. First, only recently was there some effort to create standardized benchmarks. EthProofs [30] is a portal where ZkEVM teams can join and be measured on the most important thing, the proving times of Ethereum blocks. As of writing of this work, only teams using SP1's ZkVM have their proving times listed. Other ZkVMs, are not participating in this, as well as ZkEVMs from L2s.

The SP1 from Succinct is proving Ethereum blocks averages one Ethereum block proof in around 3 minutes, with proof sizes of around 1.48MB [30].

2.4 Lattice-based cryptography

Lattice-based cryptography has emerged as a significant area of research, particularly due to its potential for constructing cryptographic primitives resistant to attacks by quantum computers. Unlike traditional public-key cryptosystems such as those based on elliptic curves, which are vulnerable to Shor's algorithm on a quantum computer, many lattice-based constructions

are believed to maintain their security in a post-quantum world. The security of these schemes often relies on the presumed hardness of certain computational problems on lattices, such as the Shortest Vector Problem (SVP) [31, 32].

A lattice L is a discrete subgroup of $\mathbb{R}^n$, typically represented as the set of all integer linear combinations of a set of linearly independent basis vectors $B = \{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m\}$ where $\mathbf{b}_i \in \mathbb{R}^n$. That is:

$$L(B) = \left\{ \sum_{i=1}^{m} x_i \mathbf{b}_i \mid x_i \in \mathbb{Z} \right\}$$

The dimension of the lattice is m. Many cryptographic constructions are built upon the difficulty of solving problems like finding the shortest non-zero vector in a lattice (SVP) or finding the lattice point closest to a given target vector (CVP), especially in high dimensions.

One of the pioneering works in this field was by Miklós Ajtai in 1996, who introduced a cryptographic construction whose security could be based on the worst-case hardness of lattice problems [31]. This provided a stronger theoretical foundation for cryptosystems using lattices.

Lattice-based cryptography offers several advantages:

- **Post-quantum security:** As mentioned, this is a primary driver for its adoption.
- **Efficiency:** Some lattice-based operations can be computationally efficient, particularly those involving matrix-vector multiplications over small integers [32].
- **Versatility:** Lattices have been used to construct a wide array of cryp-

tographic primitives, including public-key encryption, digital signatures, and importantly for this thesis, commitment schemes [32].

2.4.1 Ajtai Commitments

A commitment scheme is a cryptographic primitive that allows a party (the prover) to commit to a value while keeping it hidden from another party (the verifier), with the ability to reveal the committed value later. The scheme must satisfy two main properties:

- **Hiding:** The verifier cannot learn the committed value from the commitment itself before the reveal phase.
- **Binding:** The prover cannot change the committed value after the commitment phase.

The Ajtai commitment scheme from Ajtai's 1996 work [31], is a lattice-based commitment scheme. Its security is typically based on the hardness of the Shortest Integer Solution (SIS) problem [31, 32].

The SIS problem is defined as follows: Given a random matrix $A \in \mathbb{Z}_q^{n \times m}$ (where q is a modulus, n is the row dimension, and m is the column dimension, with $m > n \log q$), find a non-zero integer vector $\mathbf{s} \in \mathbb{Z}^m$ with small norm (i.e., $s_i \in s; ||s_i|| \leq \beta$, for some bound β) such that:

$$A\mathbf{s} = \mathbf{0} \pmod{q}$$

Finding such a short vector $\mathbf{s}$ is believed to be computationally hard for appropriate parameters.

- **Binding:** The binding property relies on the SIS assumption. If a

prover could find two different openings (e.g., $\mathbf{s}$ and $\mathbf{s}'$ where $\mathbf{s} \neq \mathbf{s}'$) for the same commitment $\mathbf{t}$, such that $A\mathbf{s} = \mathbf{t} \pmod q$ and $A\mathbf{s}' = \mathbf{t} \pmod q$, then $A(\mathbf{s} - \mathbf{s}') = \mathbf{0} \pmod q$. If $\mathbf{s} - \mathbf{s}'$ is a short non-zero vector, this would imply a solution to the SIS problem for matrix A.

- **Hiding:** The hiding property is based on the Learning With Errors (MLWE) problem or related assumptions. Informally, given A and $A\mathbf{s} \pmod q$ where $\mathbf{s}$ is short and random, it is computationally difficult to recover $\mathbf{s}$. The distribution of $A\mathbf{s} \pmod q$ is computationally indistinguishable from a uniform random vector over $\mathbb{Z}_q^n$ [33].

A useful feature of some Ajtai commitments is that the size of the commitment $\mathbf{t}$ does not grow with the dimension of the message $\mathbf{s}$, though the message components themselves must be small [33].

2.4.2 Lattices in ZK

Lattice-based cryptography presents a compelling avenue for advancing zero-knowledge proof systems, due to its strong security foundations against quantum computers. This makes them a promising candidate for future-proofing ZK solutions.

LaBRADOR: Compact Proofs for R1CS from Module-SIS

LaBRADOR, introduced by Beullens and Seiler [34], is a lattice-based proof system designed for Rank-1 Constraint Systems (R1CS). An advantage of LaBRADOR is its ability to generate very compact proof sizes, particularly for large circuits, making it more efficient in terms of proof transmission and storage compared to many other post-quantum approaches. For instance, LaBRADOR can prove knowledge of a solution for an R1CS modulo $2^{64} + 1$

with 2^{20} constraints with a proof size of only 58 KB.

However, a notable drawback of LaBRADOR is that its verifier runtime is linear in the size of the R1CS instance. This means that while the proofs are small, the verification process is not succinct, which can limit its applicability in scenarios requiring fast, sublinear verification. Despite this, LaBRADOR's compact proof generation makes it a candidate as an "outer layer" or wrapper. It can be used to prove the correctness of a (potentially much larger) proof generated by another succinct, but perhaps less compact, proof system, thereby achieving overall succinctness with post-quantum security.

Greyhound: Fast Polynomial Commitments from Lattices

Building upon the strengths of LaBRADOR, Nguyen and Seiler proposed Greyhound, an efficient polynomial commitment scheme (PCS) based on standard lattice assumptions [35]. Greyhound leverages LaBRADOR as a core component to achieve succinct proofs of polynomial evaluation.

The construction involves a three-round protocol for proving evaluations for polynomials of bounded degree N with a verifier time complexity of $O_p(\sqrt{N})$. By composing this with the LaBRADOR proof system, Greyhound obtains a succinct proof of polynomial evaluation that also has a sublinear verifier runtime. For large polynomials (e.g., degree up to $N \approx 2^{30}$), Greyhound produces evaluation proofs of approximately 53KB. This demonstrates the practical utility of using LaBRADOR to compress proofs for more complex cryptographic primitives like polynomial commitments.

LatticeFold and LatticeFold+

In the domain of folding schemes, which are techniques for building efficient recursive SNARKs, Boneh and Chen introduced LatticeFold. LatticeFold addressed the challenge of maintaining low-norm witnesses through multiple rounds of folding by employing a sumcheck-based range proof to ensure extracted witnesses remained valid [36].

More recently, Boneh and Chen proposed LatticeFold+ [37], an improved lattice-based folding protocol. LatticeFold+ enhances its predecessor in several key aspects: the prover is substantially faster (five to ten times), the verification circuit is simpler, and the folding proofs are shorter. These improvements are achieved through two main lattice techniques:

1. A new, purely algebraic range proof that is more efficient than the bit-decomposition approach used in LatticeFold.
2. The use of double commitments (commitments of commitments) to further shrink proof sizes, along with a new sumcheck-based transformation for folding statements about these double commitments.

LatticeFold+ aims to be competitive with, or even outperform, pre-quantum folding schemes like HyperNova [38] while offering plausible post-quantum security.

References

[1] *Ethereum Whitepaper — ethereum.org*. https://ethereum.org/en/whitepaper/. [Accessed 28-05-2025].

[2] Vitalik Buterin. *Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform.* https://ethereum.org/content/whitepaper/whitepaper-pdf/Ethereum_Whitepaper_-_Buterin_2014.pdf. [Accessed 28-05-2025].

[3] Vitalik Buterin et al. *Combining GHOST and Casper*. 2020. DOI: 10.48550/ARXIV.2003.03052. URL: https://arxiv.org/abs/2003.03052.

[4] *Consensus mechanisms — ethereum.org*. https://ethereum.org/en/developers/docs/consensus-mechanisms/. [Accessed 29-05-2025].

[5] *Ethereum Virtual Machine (EVM) — ethereum.org*. https://ethereum.org/en/developers/docs/evm/. [Accessed 29-05-2025].

[6] Oded Goldreich, Silvio Micali, and Avi Wigderson. "Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems". In: *Journal of the ACM (JACM)* 38.3 (1991), pp. 690–728.

[7] Amos Fiat and Adi Shamir. "How To Prove Yourself: Practical Solutions to Identification and Signature Problems". In: *Lecture Notes in*

Computer Science. Springer Berlin Heidelberg, pp. 186–194. ISBN: 9783540180470. DOI: 10.1007/3-540-47721-7_12. URL: http://dx.doi.org/10.1007/3-540-47721-7_12.

[8] Nir Bitansky et al. *From Extractable Collision Resistance to Succinct Non-Interactive Arguments of Knowledge, and Back Again*. Cryptology ePrint Archive, Paper 2011/443. 2011. URL: https://eprint.iacr.org/2011/443.

[9] - *YouTube — youtube.com*. https://www.youtube.com/watch?v=lRqnFrqpq4k. [Accessed 31-05-2025].

[10] *Native rollups—superpowers from L1 execution — ethresear.ch*. https://ethresear.ch/t/native-rollups-superpowers-from-l1-execution/21517. [Accessed 31-05-2025].

[11] *The different types of ZK-EVMs — vitalik.eth.limo*. https://vitalik.eth.limo/general/2022/08/04/zkevm.html. [Accessed 01-06-2025].

[12] Lior Goldberg, Shahar Papini, and Michael Riabzev. *Cairo – a Turing-complete STARK-friendly CPU architecture*. Cryptology ePrint Archive, Paper 2021/1063. 2021. URL: https://eprint.iacr.org/2021/1063.

[13] Jens Groth. *On the Size of Pairing-based Non-interactive Arguments*. Cryptology ePrint Archive, Paper 2016/260. 2016. URL: https://eprint.iacr.org/2016/260.

[14] Paul Valiant. "Incrementally Verifiable Computation or Proofs of Knowledge Imply Time/Space Efficiency". In: *Theory of Cryptography*. Springer Berlin Heidelberg, 1–18. ISBN: 9783540785248. DOI: 10.1007/978-3-540-78524-8_1. URL: http://dx.doi.org/10.1007/978-3-540-78524-8_1.

[15] Eli Ben-Sasson et al. *Scalable Zero Knowledge via Cycles of Elliptic Curves*. Cryptology ePrint Archive, Paper 2014/595. 2014. URL: https://eprint.iacr.org/2014/595.

[16] *plonky2/plonky2/plonky2.pdf at main · 0xPolygonZero/plonky2 — github.com*. https://github.com/0xPolygonZero/plonky2/blob/main/plonky2/plonky2.pdf. [Accessed 01-06-2025].

[17] *halo2 - The halo2 Book — zcash.github.io*. https://zcash.github.io/halo2/index.html. [Accessed 01-06-2025].

[18] Abhiram Kothapalli, Srinath Setty, and Ioanna Tzialla. *Nova: Recursive Zero-Knowledge Arguments from Folding Schemes*. Cryptology ePrint Archive, Paper 2021/370. 2021. URL: https://eprint.iacr.org/2021/370.

[19] *RISC-V Ratified Specifications — riscv.org*. https://riscv.org/specifications/ratified/. [Accessed 01-06-2025].

[20] *dev.risczero.com*. https://dev.risczero.com/proof-system-in-detail.pdf. [Accessed 01-06-2025].

[21] Eli Ben-Sasson et al. *Scalable, transparent, and post-quantum secure computational integrity*. Cryptology ePrint Archive, Paper 2018/046. 2018. URL: https://eprint.iacr.org/2018/046.

[22] Eli Ben-Sasson et al. "Fast Reed-Solomon Interactive Oracle Proofs of Proximity". In: *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*. Ed. by Ioannis Chatzigiannakis et al. Vol. 107. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018, 14:1–14:17. ISBN: 978-3-95977-076-7. DOI: 10.4230/LIPIcs.ICALP.2018.14. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ICALP.2018.14.

[23] *The RISC Zero STARK Protocol | RISC Zero Developer Docs — dev.risczero.com*. https://dev.risczero.com/proof-system/proof-system-sequence-diagram. [Accessed 01-06-2025].

[24] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. *PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge*. Cryptology ePrint Archive, Paper 2019/953. 2019. URL: https://eprint.iacr.org/2019/953.

[25] *Succinct Docs — docs.succinct.xyz*. https://docs.succinct.xyz/. [Accessed 01-06-2025].

[26] Arasu Arun, Srinath Setty, and Justin Thaler. *Jolt: SNARKs for Virtual Machines via Lookups*. Cryptology ePrint Archive, Paper 2023/1217. 2023. URL: https://eprint.iacr.org/2023/1217.

[27] Srinath Setty, Justin Thaler, and Riad Wahby. *Unlocking the lookup singularity with Lasso*. Cryptology ePrint Archive, Paper 2023/1216. 2023. URL: https://eprint.iacr.org/2023/1216.

[28] Eli Ben-Sasson et al. *DEEP-FRI: Sampling Outside the Box Improves Soundness*. Cryptology ePrint Archive, Paper 2019/336. 2019. URL: https://eprint.iacr.org/2019/336.

[29] *openvm.dev*. https://openvm.dev/whitepaper.pdf. [Accessed 01-06-2025].

[30] *Ethproofs | Ethproofs — ethproofs.org*. https://ethproofs.org/. [Accessed 02-06-2025].

[31] M. Ajtai. “Generating hard instances of lattice problems (extended abstract)”. In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*. STOC ’96. Philadelphia, Pennsylvania, USA: Association for Computing Machinery, 1996, 99–108. ISBN:

0897917855. DOI: 10.1145/237814.237838. URL: https://doi.org/10.1145/237814.237838.

[32] Yang Li, Kee Siong Ng, and Michael Purcell. *A Tutorial Introduction to Lattice-based Cryptography and Homomorphic Encryption*. 2022. DOI: 10.48550/ARXIV.2208.08125. URL: https://arxiv.org/abs/2208.08125.

[33] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plancon. *Lattice-Based Zero-Knowledge Proofs and Applications: Shorter, Simpler, and More General*. Cryptology ePrint Archive, Paper 2022/284. 2022. URL: https://eprint.iacr.org/2022/284.

[34] Ward Beullens and Gregor Seiler. *LaBRADOR: Compact Proofs for R1CS from Module-SIS*. Cryptology ePrint Archive, Paper 2022/1341. 2022. URL: https://eprint.iacr.org/2022/1341.

[35] Ngoc Khanh Nguyen and Gregor Seiler. *Greyhound: Fast Polynomial Commitments from Lattices*. Cryptology ePrint Archive, Paper 2024/1293. 2024. DOI: 10.1007/978-3-031-68403-6_8. URL: https://eprint.iacr.org/2024/1293.

[36] Dan Boneh and Binyi Chen. *LatticeFold: A Lattice-based Folding Scheme and its Applications to Succinct Proof Systems*. Cryptology ePrint Archive, Paper 2024/257. 2024. URL: https://eprint.iacr.org/2024/257.

[37] Dan Boneh and Binyi Chen. *LatticeFold+: Faster, Simpler, Shorter Lattice-Based Folding for Succinct Proof Systems*. Cryptology ePrint Archive, Paper 2025/247. 2025. URL: https://eprint.iacr.org/2025/247.

[38] Abhiram Kothapalli and Srinath Setty. *HyperNova: Recursive arguments for customizable constraint systems*. Cryptology ePrint Archive, Paper 2023/573. 2023. URL: https://eprint.iacr.org/2023/573.

Appendix A

Project task schedule

A.1 DP1

1st-4th week	Researching recursive proofs, folding schemes, lattice based cryptography
5th-7th week	Researching Ethereum's architecture
8th-9th week	Researching Beam Chain initiative
10th-11th week	Researching ZkEVMs, ZkVMs
12th-13th week	Writing analysis

A.2 DP2

1st-3rd week	Designing a ZKP proof system based on lattices
4th-6th week	Designing a ZkVM based on new lattice based ZKP system
7th-12th week	Implementing a ZkVM
13th week	Documenting design and implementation

A.3 DP3

1st-3rd week	Testing ZkEVM compiled into RISC-V in designed ZkVM
4th-6th week	Benchmarking performance, proof sizes of the ZkEVM
7th-10th week	Optimizing ZkVM
11th-13th week	Writing final version of document