Final Project Report

VIKING: A NON-MALLEABLE ZKSNARK WITHOUT TRUSTED SETUP USING R1CSLITE

Pois Bois

Gnana Prakash Punnavajhala - 2021111027

Ishwar B Balappanawar - 2021101023

Shrikara A - 2021101058

Vansh Pravin Marda - 2021101089

Yash Prashant Adivarekar - 2021101008

Contents

Introduction

Motivation

From [Set20], Zero-knowledge succinct non-interactive arguments of knowledge (zkSNARKs) [BCCT12, Gro16] enable a computationally-bounded prover to convince the membership of a problem instance in an NP language by producing a proof without revealing anything besides the validity of the statement. Furthermore, the proof size and the verifier's costs are sub-linear in the size of the statement. The motivation to design zkSNARKs is because they enable many applications that involve various forms of delegation of computation for scalability or privacy [SCG+14, BBCG+19, BCG+20].

There are many approaches to construct such arguments in the literature, starting with the work of Kilian [Kil92] who provided the first construction of a succinct interactive argument protocol by employing probabilistically checkable proofs (PCPs) [ALM+98, AS98, BFLS91] in conjunction with Merkle trees. Micali made a similar protocol non-interactive in the random oracle model, thereby obtaining the first zkSNARK. Unfortunately, the underlying PCP machinery remains extremely expensive for the prover and the verifier—despite foundational advances. Thus, the first works with an explicit motivation to make proof systems practical refine and implement interactive protocols of Ishai et al. and Goldwasser et al., which do not require asymptotically-efficient PCPs. The principal downside is that they achieve practicality for only a restricted class of NP statements.

Gennaro, Gentry, Parno, and Raykova (GGPR) address the above issue with a new characterization of NP called quadratic arithmetic programs (QAPs). By building on the work of Ishai et al., Groth , and Lipmaa, GGPR construct a zkSNARK for R1CS in which the prover's running time is O(n log n), the size of a proof is O(1), and the verifier incurs O(|io|) computation to verify a proof, where n is the size of the statement, and io denotes the public input and output. Unfortunately, GGPR's zkSNARK requires a per-statement trusted setup that produces an O(n)-sized structured common reference string and the trapdoor used in the setup process must be kept secret to ensure soundness. Relying on such a trusted setup is often infeasible, especially for applications that do not have trusted authorities. There exist several advances atop GGPR, but they retain a trusted setup [BSCG+13, BSCTV14], or require interaction.

The above state of affairs has motivated another class of works, called transparent zkSNARKs, that aim to eliminate the requirement of a trusted setup. They prove security in the random oracle model, which is acceptable in practice. First, Hyrax extends a line of work that refines the doubly-efficient interactive proofs (IPs) of Goldwasser et al.. Second, STARK and Aurora build on interactive oracle proofs (IOPs). Third, Ligero builds on the "MPC in the head" paradigm. Fourth, Bulletproofs builds on the work of Bootle et al.

Unfortunately, they face the following problems.

- The computational model of Hyrax is layered arithmetic circuits, where the verifier's costs and the proof sizes scale linearly in the depth of the circuit. Converting an arbitrary circuit into a layered form can increase its size quadratically, so Hyrax is restricted to low-depth circuits. Also, Hyrax achieves sub-linear verification costs only for circuits with a uniform structure (e.g., data-parallel circuits).
- STARK requires circuits with a sequence of identical sub-circuits, otherwise it does not achieve sub-linear verification costs. Any circuit can be converted to this form, but the transformation increases circuit sizes by 10–1000x, which translates to a similar factor increase in the prover's costs.
- Ligero, Bulletproofs, and Aurora incur O(n) verification costs.

Applications

The increasing use of zkSNARKs in real world systems such as blockchains necessitates minimizing cost to create a proof, specifically when used in edge devices with limited computational resources. Simulation extractability and its implied non malleability are needed to ensure robustness, such as in voting systems (Ex - Helios[BPW12] and SwissPost [HLPT20] voting systems were compromised due to malleability). Removing a trusted setup is needed to ensure that false proofs cannot be created if the trusted setup is compromised. ZCash[Z.C] recently removed their trusted setup by shifting to Halo [BGH19].

Literature Survey

Spartan: Efficient and general-purpose zkSNARKs without trusted setup

- Most existing zkSNARK protocols rely on trusted setups which are commonly accused of generating toxic waste and are potential targets to undermine the transparency and security of the system. Hence, it is enticing to eliminate them from the pipeline
- Spartan introduces a new family of zkSNARKs that achieve sub-linear verification costs for arbitrary NP statements without relying on trusted setups. Spartan offers zkSNARKs with a time-optimal prover, an issue faced by nearly all zkSNARKs in the literature.
- Spartan provides a public-coin succinct interactive argument of knowledge for NP using the sum-check protocol composed with compactly encoded R1CS instances and computational commitments.
- Zero knowledge and Non-interactibility are achieved via existing compilers and Fiat-Shamir transform since the argument is public-coin

Counting vampires: From univariate sumcheck to updatable ZK-SNARK

- Most existing updatable and universal zkSNARKs use sumchecks with polynomial commitment schemes like Aurora which has the complexity of 2 field elements
- This paper introduces Count, a sumcheck based on inner product commitment schemes like ILV inner-product scheme, which lets them get away with using just 1 field element
- Most known universal and updatable SNARKs use commitment schemes modellled by AHPs/PHPs like the KZG scheme which depend on consequent SRSs (those with no gaps)
- This paper uses the ILV scheme which can handle non-consequent SRSs, (can help build protocols like broadcast encryption) to propose the Count sumcheck argument
- They finally come up with a zkSNARK called Vampire that utilizes Count and it works for the R1CSLite constraint system

From Polynomial IOP and Commitments to Non-malleable zkSNARKs

- Most zkSNARKs satisfy knowledge soundness but not simulation extractability (SE). SE implies non-malleability.
- This paper introduces a relaxed notion of SE called policy based SE (ϕ-SE) for polynomial commitments
- It proves that the KZG commitment scheme satisfies this relaxed notion
- It proves that this is sufficient to compile a SE zkSNARK
- It proves the SE of existing zkSNARKs - Plonk, Basilisk, variants of Marlin and Lunar
- Any future zkSNARK that satisfies this relaxed notion will also be SE

Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS

- The paper introduces the concept of preprocessing zkSNARKs using Universal and Updatable SRS (Structured Reference Strings). This innovation aims to enhance the efficiency and scalability of zkSNARKs by providing a more versatile and dynamic setup, allowing for universal use across different applications and enabling updates to the SRS without compromising security.
- The paper makes use of Algebraic Holographic Proofs and Extractable Polynomial commitment to form the basis of Universal and Updatable SRS for zkSNARKS.
- The Marlin protocol promises improved performance and adaptability for a wide range of practical applications.

Scalable Multi-party Computation for zk-SNARK Parameters in the Random Beacon Model

- zk-SNARKs promise privacy-preserving proofs, but relying on a TTP contradicts this goal.
- The paper by Bowe et al. aims to reimagine how zk-SNARK parameters are generated, moving away from a centralized approach with a trusted third party (TTP) towards a secure and scalable multi-party solution.
- Paper Guarantees security in a distributed setting without a TTP requires careful consideration. The paper introduces the random beacon model, where a trusted source of randomness ensures security even if some participants are compromised.
- This integrated randomness strengthens security even if the adversary has limited influence over the beacon, ensuring the integrity of the generated parameters.

Problem Statement

Problem

The setting of the problem statement we intend to tackle is as follows:

Novel zero-knowledge Succinct Non-interactive ARguments of Knowledge (zkSNARKs) without trusted setup for Rank-1 Constraint System Lite (R1CSLite), an NP-Complete language that can be used to generalize arithmetic satisfiability more efficiently than existing implementations and proposals.

These arguments for knowledge can be used by a prover to convince a verifier of his knowledge about a witness to the NP satisfiability problem.

Proposal

We intend to generalize of arithmetic satisfiability by employing the R1CSLite language which is NP-Complete. Subsequently, we aim to emulate the procedures employed by Spartan to achieve a zkSNARK solution which does not require a trusted setup.

Novelty

The novelty arises from the fact that while people have attempted to use R1CSLite representations for their arguments of knowledge like [LSZ22], [CFF^{+}21], all of them rely on a trusted setup leading to the limitations and criticisms of existing zkSNARK protocols, particularly the reliance on trusted setups. Trusted setups introduce concerns regarding the generation and disposal of toxic waste, which undermines the transparency and security of the system. By eliminating the need for trusted setups, we aim to address these concerns and provide a more robust solution for proving NP statements. Additionally, the desire to achieve sub-linear verification costs and a time-optimal prover reflects a broader goal of improving the efficiency and usability of zkSNARKs, making them more practical for real-world applications

Existing State of the Art

We search for the existing state of the art for proved simulation extractable (or at least non-malleable) zkSNARKs which are transparent, i.e. do not require a trusted setup. We find already that the number of transparent zkSNARKs are low and zkSNARKs which are provably simulation extractable are for zkSNARKs with trusted setups. In this context we find:

- Spartan and Bulletproofs are Simulation-Extractable (for Free!) [DG23]: Proves that two transparent, discrete-log-based zkSNARKs, Spartan and Bullet-proofs, are simulation-extractable (SIM-EXT) in the random oracle model (ROM) if the discrete logarithm (DLOG) assumption holds in the underlying group. Since these assumptions are required to prove standard security properties for Spartan and Bulletproofs, their results show that SIM-EXT is "for free" with these schemes. Their result is the first SIM-EXT proof for Spartan and encompasses both linear- and sublinear-verifier variants. Their result for Bulletproofs encompasses both the aggregate range proof and arithmetic circuit variants, and is the first to not rely on the algebraic group model (AGM).

- What Makes Fiat–Shamir zkSNARKs (Updatable SRS) Simulation Extractable? [GKK+22]: They show that three popular universal zero-knowledge SNARKs (Plonk, Sonic, and Marlin) are updatable SRS simulation extractable NIZKs and signatures of knowledge (SoK) out-of-the-box avoiding any compilation overhead. They generalize results for the Fiat–Shamir (FS) transformation, which turns interactive protocols into signature schemes, non-interactive proof systems, or SoK in the random oracle model (ROM).

- Fiat–Shamir Bulletproofs are Non-Malleable (in the Algebraic Group Model)[GOP+22]: Prior to this work, there was no evidence that malleability attacks were not possible against Fiat-Shamir Bulletproofs. Malleability attacks can lead to very severe vulnerabilities, as they allow an adversary to forge proofs re-using or modifying parts of the proofs provided by the honest parties. This paper shows for the first time that Bulletproofs (or any other similar multi-round proof system satisfying some form of weak unique response property) achieve simulation-extractability in the algebraic group model, which implies that Fiat-Shamir Bulletproofs are non-malleable

Preliminaries

Some adapted from [Set20] We use $\mathbb{F}$ to denote a finite field (e.g., the prime field $\mathbb{F}_p$ for a large prime p) and λ to denote the security parameter. We use negl (λ) to denote a negligible function in λ. Throughout the paper, the depicted asymptotics depend on λ, but we elide this for brevity. We use "PPT algorithms" to refer to probabilistic polynomial time algorithms.

Problem instances in R1CS

Recall that for any problem instance x, if x is in an NP language $\mathcal{L}$, there exists a witness w and a deterministic algorithm Sat such that:

$$\mathrm{Sat}_{\mathcal{L}}(x, w) = \begin{cases} 1 & \text{if } x \in \mathcal{L} \\ 0 & \text{otherwise} \end{cases}$$

Alternatively, the set of tuples of the form $\langle \mathbb{x}, w \rangle$ form a set of NP relations. The subset of those for which $\mathrm{Sat}_{\mathcal{L}}(\mathbb{x}, w) = 1$ are called satisfiable instances, which we denote as: $\mathcal{R}_{\mathcal{L}} = \{\langle \mathbb{x}, w \rangle : \mathrm{Sat}_{\mathcal{L}}(\mathbb{x}, w) = 1\}$.

As an NP-complete language, we focus on the rank-1 constraint satisfiability (R1CS). As noted earlier, R1CS is a popular target for compiler toolchains that accept applications expressed in high-level languages. R1CS is implicit in the QAPs of GGPR, but it is used with (and without) QAPs in subsequent works.

Definition 2.1 (R1CS instance). An R1CS instance is a tuple $(\mathbb{F}, A, B, C, io, m, n)$, where io denotes the public input and output of the instance, $A, B, C \in \mathbb{F}^{m \times m}$, where $m \geq |io| + 1$ and there are at most n non-zero entries in each matrix.

Note that matrices A, B, C are defined to be square matrices for conceptual simplicity. Below, we use the notation $z = (x, y, z)$ (where each of x, y, z is a vector over $\mathbb{F}$) to mean that z is a vector that concatenates the three vectors in a natural way. WLOG, we assume that $n = O(m)$ throughout the paper.

Definition 2.2 (R1CS). An R1CS instance $(\mathbb{F}, A, B, C, io, m, n)$ is said to be satisfiable if there exists a witness $w \in \mathbb{F}^{m-|io|-1}$ such that $(A \cdot z) \circ (B \cdot z) = (C \cdot z)$, where $z = (io, 1, w)$, $\cdot$ is the matrix-vector product, and $\circ$ is the Hadamard (entry-wise) product.

Note that R1CS generalizes arithmetic circuit satisfiability because the entries in matrices A, B, C can be used to encode addition and multiplication gates over $\mathbb{F}$. Furthermore, they can be used to encode a class of degree- 2 constraints of the form $L(z) \cdot R(z) = O(z)$, where L, R, O are degree-1 polynomials over variables that take values specified by $z = (io, 1, w)$. In other words, R1CS supports arbitrary fan-in addition gates, and multiplication gates that verify arbitrary bilinear relations over the entire z.

Definition 2.3. For an R1CS instance $\mathbb{x} = (\mathbb{F}, A, B, C, io, m, n)$ and a purported witness $w \in \mathbb{F}^{m-|io|-1}$, we define:

$$\mathrm{Sat}_{\mathrm{R1CS}}(\mathbb{x}, w) = \begin{cases} 1 & (A \cdot z) \circ (B \cdot z) = (C \cdot z) \\ 0 & \text{otherwise} \end{cases}$$

The set of satisfiable R1CS instances can be denoted as:

$$\mathcal{R}_{\mathrm{R1}CS} = \{\langle(\mathbb{F}, A, B, C, io, m, n), w\rangle : \mathrm{Sat}_{\mathrm{R1CS}}((\mathbb{F}, A, B, C, io, m, n), w) = 1\}$$

Definition 2.4. For a given R1CS instance $\mathbb{x} = (\mathbb{F}, A, B, C, io, m, n)$, the NP statement that $\mathbb{x}$ is satisfiable (i.e., $\langle \mathbb{x}, \cdot\rangle \in \mathcal{R}_{\mathrm{R1CS}}$) is of size $O(n)$.

Succinct interactive arguments of knowledge

Let $\langle \mathcal{P}, \mathcal{V}\rangle$ denote a pair of PPT interactive algorithms and Setup denotes an algorithm that outputs public parameters pp given as input the security parameter λ.

Definition 2.5. A protocol between a pair of PPT algorithms $\langle \mathcal{P}, \mathcal{V}\rangle$ is called a publiccoin succinct interactive argument of knowledge for a language $\mathcal{L}$ if:

- **Completeness.** For any problem instance $x \in \mathcal{L}$, there exists a witness w such that for all $r \in \{0,1\}^*, \Pr\{\langle \mathcal{P}(pp, w), \mathcal{V}(pp, r)\rangle(\mathbb{x}) = 1\} \geq 1 - \mathrm{negl}(\lambda)$.

- **Soundness.** For any non-satisfiable problem instance $\mathbb{x}$, any PPT prover $\mathcal{P}^*$, and for all $w, r \in \{0,1\}^*, \Pr\left\{\langle \mathcal{P}^*(pp, w), \mathcal{V}(pp, r)\rangle(\mathbb{x}) = 1\right\} \leq \mathrm{negl}(\lambda)$.

- **Knowledge soundness.** For any PPT adversary $\mathcal{A}$, there exists a PPT extractor $\mathcal{E}$ such that for any problem instance $\mathbb{x}$ and for all $w, r \in \{0,1\}^*$, if
$\Pr\{\langle \mathcal{A}(pp, w), \mathcal{V}(pp, r)\rangle(\mathbb{x}) = 1\} \geq \mathrm{negl}(\lambda)$, then
$\Pr\left\{\mathrm{Sat}_{\mathcal{L}}(\mathbb{x}, w') = 1 \mid w' \leftarrow \mathcal{E}^{\mathcal{A}}(pp, \mathbb{x})\right\} \geq \mathrm{negl}(\lambda)$.

- **Succinctness.** The total communication between $\mathcal{P}$ and $\mathcal{V}$ is sub-linear in the size of the NP statement $x \in \mathcal{L}$.

- **Public coin.** V's messages are chosen uniformly at random.

Definition 2.6 (Witness-extended emulation). An interactive argument (Setup, $\mathcal{P}, \mathcal{V}$) for $\mathcal{L}$ has witness-extended emulation if for all deterministic polynomial time programs $\mathcal{P}^*$ there exists an expected polynomial time emulator E such that for all non-uniform polynomial time adversaries A and all $z_{\mathcal{V}} \in \{0,1\}^*$, the following probabilities differ by at most $negl(\lambda)$:

$$\Pr\left\{pp \leftarrow \mathrm{Setup}\left(1^\lambda\right); (\mathbb{x}, z_{\mathcal{P}}) \leftarrow A(pp); t \leftarrow \mathrm{tr}\left\langle \mathcal{P}^*\left(z_{\mathcal{P}}\right), \mathcal{V}\left(z_{\mathcal{V}}\right)\right\rangle(\mathbb{x}) : A(t) = 1\right\}$$

and

$$\Pr\{pp \leftarrow \mathrm{Setup}\left(1^\lambda\right); (\mathbb{x}, z_{\mathcal{P}}) \leftarrow A(pp); (t, w) \leftarrow E^{\mathcal{P}^*(z_{\mathcal{P}})}(\mathbb{X}) : A(t) = 1 \wedge$$
$$\text{if } t \text{ is an accepting transcript then} \operatorname{Sat}_{\mathcal{L}}(\mathbb{x}, w) = 1\}$$

where tr denotes the random variable that corresponds to the transcript of the interaction between $\mathcal{P}^*$ and $\mathcal{V}$.

Definition 2.7. An interactive argument (Setup, $\mathcal{P}, \mathcal{V}$) for $\mathcal{L}$ is computational zero knowledge if for every PPT interactive machine $\mathcal{V}^*$, there exists a PPT algorithm S called the simulator, running in time polynomial in the length of its first input such that for every problem instance $\mathbf{x} \in \mathcal{L}, w \in \mathcal{R}_{\mathrm{x}}$, and $z \in \{0,1\}^*$, the following holds when the distinguishing gap is considered as a function of $|\mathbf{x}|$:

$$\text{View}\left(\langle \mathcal{P}(w), \mathcal{V}^*(z)\rangle(\mathbf{x})\right) \approx_c S(\mathbf{x}, z)$$

where View $(\langle \mathcal{P}(w), \mathcal{V}^*(z)\rangle(\mathbf{x}))$ denotes the distribution of the transcript of interaction between $\mathcal{P}$ and $\mathcal{V}^*$, and $\approx_c$ denotes that the two quantities are computationally indistinguishable. If the statistical distance between the two distributions is negligible then the interactive argument is said to be statistical zero-knowledge. If the simulator is allowed to abort with probability at most 1/2, but the distribution of its output conditioned on not aborting is identically distributed to View $(\langle \mathcal{P}(w), \mathcal{V}^*(z)\rangle(\mathbf{x}))$, then the interactive argument is called perfect zero-knowledge.

Problem instances in R1CSLite

A satisfiable R1CSLite instance is a tuple $(\mathbb{F}, n, m, l, L, R)$ where $L, R \in \mathbb{F}^{n\times n}$, $max\{||L||, ||R||\} \leq m$, the first l rows of R are $(-1, 0, \ldots 0) \in \mathbb{F}^{1\times n}$, $\mathbf{x} \in \mathbb{F}^{l-1}$, $w \in \mathbb{F}^{n-l}$ and for $z = (1, x, w)$ we have

$$(L \cdot z) \circ (R \cdot z) = z$$

where $\cdot$ is the matrix product and $\circ$ is the Hadamard product

To compare the efficiency of the R1CS and R1CSLite constraint systems that result from formulating the respective constraints as arithmetic circuit satisfiability problems to the same circuit C, let $\hat{L}, \hat{R}, \hat{O}$ and L, R be the resulting R1CS and R1CSLite matrices respectively. From the theorems statements it's clear that R1CS matrices have $l_{in} + 1$ less rows than R1CSLite ones.

Next, to analyze their densities, for $\mathbf{c} = (1, \mathbf{x}, \mathbf{y}, \mathbf{c}')$, the satisfiability can be expressed as the following constraints:

$$\begin{cases} (L' \cdot c) \circ (R' \cdot c) = [0_{N\times l}|I] \cdot c \\ L_{out} \cdot c = 0 \end{cases} \qquad \text{with } L' = \begin{pmatrix} L_{l+1} \\ \vdots \\ L_n \end{pmatrix}, \quad R' = \begin{pmatrix} R_{l+1} \\ \vdots \\ R_n \end{pmatrix}$$

where the i-th row of $\mathbf{L}_{out}$ checks that the i-th output $\mathbf{y_i}$ is correctly obtained from a linear subcircuit with inputs from $(1, \mathbf{x}, \mathbf{c}')$.

Then one can set

$$\hat{L} = \begin{pmatrix} L' \\ L_{out} \end{pmatrix}, \quad \hat{R} = \begin{pmatrix} R' \\ (1,0,\ldots,0) \\ \vdots \\ (1,0,\ldots,0) \end{pmatrix}, \quad \hat{O} = \begin{pmatrix} 0 \mid I \\ 0_{l_{out}\times n} \end{pmatrix}$$

For R1CS:

$$||\hat{L}|| = ||L'|| + ||L_{out}||, \quad ||\hat{R}|| = ||R'|| + l_{out}, \quad ||\hat{O}|| = N$$

whereas for R1CSLite:

$$||L|| = ||L'|| + ||L_{out}|| + ||I_{in}|| + 1, \quad ||\hat{R}|| = ||R'|| + l,$$

Basically, the two R1CSLite matrices have each $l_{in} + 1$ more entries than their R1CS counterparts. On considering the total size of the constraint system, it can be observed that

$$||L|| + ||R|| < ||\hat{L}|| + ||\hat{R}|| + ||\hat{O}||$$

holds as long as $N > 2(l_{in} + 1)$, which is likely to be the case.

Polynomials and low-degree extensions

We recall a few basic facts about polynomials:

- A polynomial $\mathcal{G}$ over $\mathbb{F}$ is an expression consisting of a sum of monomials where each monomial is the product of a constant (from $\mathbb{F}$) and powers of one or more variables (which take values from $\mathbb{F}$); all arithmetic is performed over $\mathbb{F}$.
- The degree of a monomial is the sum of the exponents of variables in the monomial; the degree of a polynomial $\mathcal{G}$ is the maximum degree of any monomial in $\mathcal{G}$. Furthermore, the degree of a polynomial $\mathcal{G}$ in a particular variable x_i is the maximum exponent that x_i takes in any of the monomials in $\mathcal{G}$.
- A multivariate polynomial is a polynomial with more than one variable; otherwise it is called a univariate polynomial.

Definition 2.8 (Multilinear polynomial). A multivariate polynomial is called a multilinear polynomial if the degree of the polynomial in each variable is at most one.

Definition 2.9 (Low-degree polynomial). A multivariate polynomial $\mathcal{G}$ over a finite field $\mathbb{F}$ is called low-degree polynomial if the degree of $\mathcal{G}$ in each variable is exponentially smaller than $|\mathbb{F}|$.

Low-degree extensions (LDEs). Suppose $g : \{0,1\}^m \to \mathbb{F}$ is a function that maps m bit elements into an element of $\mathbb{F}$. A polynomial extension of g is a low-degree m-variate polynomial $\widetilde{g}(\cdot)$ such that $\widetilde{g}(x) = g(x)$ for all $x \in \{0,1\}^m$.

A multilinear polynomial extension (or simply, a multilinear extension, or MLE) is a low-degree polynomial extension where the extension is a multilinear polynomial (i.e., the degree of each variable in $\widetilde{g}(\cdot)$ is at most one). Given a function $Z : \{0,1\}^m \to \mathbb{F}$, the multilinear extension of $Z(\cdot)$ is the unique multilinear polynomial $\widetilde{Z} : \mathbb{F}^m \to \mathbb{F}$. It can be computed as follows.

$$\begin{aligned}\widetilde{Z}\left(x_1, \ldots, x_m\right) &= \sum_{e \in \{0,1\}^m} Z(e) \cdot \prod_{i=1}^{m} \left(x_i \cdot e_i + (1 - x_i) \cdot (1 - e_i)\right) \\ &= \sum_{e \in \{0,1\}^m} Z(e) \cdot \widetilde{\mathrm{eq}}(x, e) \\ &= \left\langle \left(Z(0), \ldots, Z\left(2^m - 1\right)\right), \left(\widetilde{\mathrm{eq}}(x, 0), \ldots, \widetilde{\mathrm{eq}}\left(x, 2^m - 1\right)\right) \right\rangle\end{aligned}$$

Note that $\tilde{\mathrm{eq}}\ (x, e) = \prod_{i=1}^{m} \left(e_i \cdot x_i + (1 - e_i) \cdot (1 - x_i)\right)$, which is the MLE of the following function:

$$\mathrm{eq}(x, e) = \begin{cases} 1 & \text{if } x = e \\ 0 & \text{otherwise} \end{cases}$$

For any $r \in \mathbb{F}^m, \widetilde{Z}(r)$ can be computed in $O\left(2^m\right)$ operations in $\mathbb{F}$.

Dense representation for multilinear polynomials. Since the MLE of a function is unique, it offers the following method to represent any multilinear polynomial. Given a multilinear polynomial

$\mathcal{G}(\cdot): \mathbb{F}^m \rightarrow \mathbb{F}$, it can be represented uniquely by the list of evaluations of $\mathcal{G}(\cdot)$ over the Boolean hypercube $\{0,1\}^m$ (i.e., a function that maps $\{0,1\}^m \rightarrow \mathbb{F}$). We denote such a representation of $\mathcal{G}$ as DenseRepr $(\mathcal{G})$.

Lemma 2.1. Iffor any $x \in \{0,1\}^m, \mathcal{G}(x)=0$ then DenseRepr $(\mathcal{G})$ does not have to include an entry for x.

Proof. Recall the closed-form expression for evaluating $\mathcal{G}(\cdot)$ at $(r_1, \ldots, r_m) \in \mathbb{F}^m$: $\mathcal{G}(r_1, \ldots, r_m) = \sum_{x \in \{0,1\}^m} \mathcal{G}(x) \cdot \prod_{i=1}^m (r_i \cdot x_i + (1-r_i) \cdot (1-x_i))$. Observe that if for any $x \in \{0,1\}^m, \mathcal{G}(x)=0, x$ does not contribute to $\mathcal{G}(r)$ for any $r \in \mathbb{F}^m$.

Definition 2.10. A multilinear polynomial $\mathcal{G}: \mathbb{F}^m \rightarrow \mathbb{F}$ is a sparse multilinear polynomial if $\mid$ DenseRepr $(\mathcal{G}) \mid$ is sub-linear in $O(2^m)$. Otherwise, it is a dense multilinear polynomial.

As an example, suppose $\mathcal{G}: \mathbb{F}^{2s} \rightarrow \mathbb{F}$. Suppose $\mid$ DenseRepr $(\mathcal{G}) \mid = O(2^s)$, then $\mathcal{G}(\cdot)$ is a sparse multilinear polynomial because $O(2^s)$ is sublinear in $O\left(2^{2s}\right)$.

A polynomial commitment scheme for multilinear polynomials

We adopt our definitions from Bünz et al. where they generalize the definition of Kate et al. to allow interactive evaluation proofs. We also borrow their notation: in a list of arguments or returned tuples, variables before the semicolon are public and the ones after are secret; when there is no secret information, semicolon is omitted.

WLOG, below, when algorithms accept as input a multilinear polynomial, they use the dense representation of multilinear polynomials (§2.3).

A polynomial commitment scheme for multilinear polynomials is a tuple of four protocols PC = (Setup, Commit, Open, Eval):

- $pp \leftarrow \text{Setup}\left(1^\lambda, \mu\right)$: takes as input μ (the number of variables in a multilinear polynomial); produces public parameters pp.
- $(\mathcal{C}; \mathcal{S}) \leftarrow \text{Commit}(pp; \mathcal{G})$: takes as input a μ-variate multilinear polynomial over a finite field $\mathcal{G} \in \mathbb{F}[\mu]$; produces a public commitment $\mathcal{C}$ and a secret opening hint $\mathcal{S}$.
- $b \leftarrow \text{Open}(pp, \mathcal{C}, \mathcal{G}, \mathcal{S})$: verifies the opening of commitment $\mathcal{C}$ to the μ-variate multilinear polynomial $\mathcal{G} \in \mathbb{F}[\mu]$ with the opening hint $\mathcal{S}$; outputs a $b \in \{0,1\}$.
- $b \leftarrow \text{Eval}(pp, \mathcal{C}, r, v, \mu; \mathcal{G}, \mathcal{S})$ is an interactive public-coin protocol between a PPT prover $\mathcal{P}$ and verifier $\mathcal{V}$. Both $\mathcal{V}$ and $\mathcal{P}$ hold a commitment $\mathcal{C}$, the number of variables μ, a scalar $v \in \mathbb{F}$, and $r \in \mathbb{F}^\mu . \mathcal{P}$ additionally knows a μ-variate multilinear polynomial $\mathcal{G} \in \mathbb{F}[\mu]$ and its secret opening hint $\mathcal{S} . \mathcal{P}$ attempts to convince $\mathcal{V}$ that $\mathcal{G}(r)=v$. At the end of the protocol, $\mathcal{V}$ outputs $b \in \{0,1\}$.

Definition 2.11. A tuple of four protocols (Setup, Commit, Open, Eval) is an extractable polynomial commitment scheme for multilinear polynomials over a finite field $\mathbb{F}$ if the following conditions hold.

- **Completeness.** For any m-variate multilinear polynomial $\mathcal{G} \in \mathbb{F}[\mu]$,

$$\operatorname{Pr}\left\{\begin{array}{c} pp \leftarrow \text{Setup}\left(1^\lambda, \mu\right);(\mathcal{C}, \mathcal{S}) \leftarrow \text{Commit}(pp; \mathcal{G}): \\ \text{Eval}(pp, \mathcal{C}, r, v, \mu; \mathcal{G}, \mathcal{S})=1 \wedge v=\mathcal{G}(r) \end{array}\right\} \geq 1-\operatorname{negl}(\lambda)$$

- **Binding.** For any PPT adversary $\mathcal{A}$, size parameter $\mu \geq 1$,

$$\operatorname{Pr}\left\{\begin{array}{c} pp \leftarrow \text{Setup}\left(1^\lambda, m\right);\left(\mathcal{C}, \mathcal{G}_0, \mathcal{G}_1, \mathcal{S}_0, \mathcal{S}_1\right)=\mathcal{A}(pp); \\ b_0 \leftarrow \text{Open}\left(pp, \mathcal{C}, \mathcal{G}_0, \mathcal{S}_0\right); b_1 \leftarrow \text{Open}\left(pp, \mathcal{C}, \mathcal{G}_1, \mathcal{S}_1\right): \\ b_0=b_1 \neq 0 \wedge \mathcal{G}_0 \neq \mathcal{G}_1 \end{array}\right\} \leq \operatorname{negl}(\lambda)$$

- **Knowledge soundness.** Eval is a public-coin succinct interactive argument of knowledge with witness-extended emulation (Definition 2.6) for the following NP relation given $pp \leftarrow \text{Setup}\left(1^\lambda, \mu\right)$ for a size parameter on the number of variables μ :

$$\mathcal{R}_{\text{Eval}}\ (pp) = \{\langle(\mathcal{C}, r, v), (\mathcal{G}, \mathcal{S})\rangle : \mathcal{G} \in \mathbb{F}[\mu] \wedge \mathcal{G}(r) = v \wedge \text{Open}(pp, \mathcal{C}, \mathcal{G}, \mathcal{S}) = 1\}$$

Definition 2.12. An extractable polynomial commitment scheme (Setup, Commit, Open, Eval) provides hiding commitments if for all PPT adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$:

$$\left| 1 - 2 \cdot \Pr\left\{ \begin{array}{c} b = \bar{b} : \\ pp \leftarrow \text{Setup}\left(1^\lambda, m\right); \\ (\mathcal{G}_0, \mathcal{G}_1, st) = \mathcal{A}_0(pp); \\ b \leftarrow R\{0,1\}; \\ (\mathcal{C}, \mathcal{S}) \leftarrow \text{Commit}\left(pp, \mathcal{G}_b\right); \\ \bar{b} \leftarrow \mathcal{A}_1(st, \mathcal{C}) \end{array} \right\} \right| \leq \text{negl}(\lambda)$$

If the above holds for all algorithms, then the commitment is statistically hiding.

Definition 2.13. An extractable polynomial commitment scheme (Setup, Commit, Open, Eval) with hiding commitments (Definition 2.12) is zero-knowledge if Eval is a public-coin succinct interactive argument of knowledge with witness-extended emulation (Definition 2.6) and zero-knowledge (Definition 2.7) for the following NP relation given $pp \leftarrow \text{Setup}\left(1^\lambda, \mu\right)$ for a size parameter on the number of variables μ :

$$\mathcal{R}_{\text{Eval}}\ (pp) = \{\langle(\mathcal{C}, r, v), (\mathcal{G}, \mathcal{S})\rangle : \mathcal{G} \in \mathbb{F}[\mu] \wedge \mathcal{G}(r) = v \wedge \text{Open}(pp, \mathcal{C}, \mathcal{G}, \mathcal{S}) = 1\}$$

Sumcheck

- Consider a μ variate polynomial $\mathcal{G}$ where the degree of every variable is at most l. The verifier $\mathcal{V}_{\mathcal{SC}}$ wants to check if the claim of an untrusted prover $\mathcal{P}_{\mathcal{SC}}$:

$$T = \sum_{x_1 \in \{0,1\}} \sum_{x_2 \in \{0,1\}} \cdots \sum_{x_\mu \in \{0,1\}} \mathcal{G}(x_1, x_2, \ldots x_\mu)$$

- A probabilistic guarantee version involves μ rounds of interaction and finally $\mathcal{V}_{\mathcal{SC}}$ outputs $b \in \{0,1\}$. The cost to $\mathcal{V}_{\mathcal{SC}}$ is to evaluate $\mathcal{G}$ at a random point $r \in \mathbb{F}^\mu$

Sumchecks satisfy:

- **Completeness** If $T = \sum_{x \in \{0,1\}^\mu} \mathcal{G}(x)$, then for a correct $\mathcal{P}_{SC}$ $\forall r \in \{0,1\}^*$, $\Pr\{(\mathcal{P}_{SC}(\mathcal{G}), \mathcal{V}_{SC}(r))(\mu, \ell, T) = 1\} = 1$.
- **Soundness** If $T \neq \sum_{x \in \{0,1\}^\mu} \mathcal{G}(x)$, then for any $\mathcal{P}^*_{SC}$ and $\forall r \in \{0,1\}^*$, $\Pr\{(\mathcal{P}^*_{SC}(\mathcal{G}), \mathcal{V}_{SC}(r))(\mu, \ell, T) = 1\} \leq \ell \cdot \mu / |\mathbb{F}|$.
- **Succinctness** The communication between $\mathcal{P}_{SC}$ and $\mathcal{V}_{SC}$ is $O(\mu \cdot \ell)$ elements of $\mathbb{F}$.

Fiat-Shamir Transform

Let $\Pi = (\texttt{Setup}, \mathcal{P}, \mathcal{V})$ be a public-coin $(2r+1)$-message interactive argument of knowledge.
$tr = (a_1, c_1, \ldots, a_r, c_r, a_{r+1})$ is the transcript. Turn Π into a non-interactive protocol Π_{FS} in the ROM, where:

- $\texttt{Setup}_{FS}(\text{pp}_{\mathcal{G}})$ is the same as $\texttt{Setup}(\texttt{pp}_{\mathcal{G}})$
- The prover $\mathcal{P}_{FS}$, on input (pp, x, w), invokes $\mathcal{P}(x, w)$, and instead of asking the verifier for challenge c_i, queries the random oracle to get $c_i = H(\text{pp}, x, a_1, \ldots, a_i)$ $\forall i = 1, \ldots, r$.
- $\mathcal{P}_{FS}$ outputs a non-interactive proof $\pi = (a_1, \ldots, a_r, a_{r+1})$.
- The verifier $\mathcal{V}_{FS}$, on input (pp, x, π), derives challenges c_i's by querying the random oracle as $\mathcal{P}_{FS}$ does, then runs $\mathcal{V}(\text{pp}, x, (a_1, c_1, \ldots, a_r, c_r, a_{r+1}))$ and outputs what $\mathcal{V}$ outputs.

Π_{FS} satisfies completeness, knowledge soundness, zero knowledge.

Simulation Extractability (SE)

Adapted from [DG23]

Definition of Simulation Extractability (SE)

Let $\Pi = (\texttt{Setup}, P, V)$ be a public-coin zero-knowledge interactive argument with associated $\texttt{NIZK}$ $\Pi_{FS} = (\texttt{Setup}, P_{FS}, V_{FS})$. We say Π_{FS} satisfies simulation extractability ($\texttt{SIM-EXT}$) with respect to a simulator S if there exists an efficient simulator-extractor $\mathcal{E}$ such that for every $\texttt{PPT}$ adversary P^*, the following probability is negligible in λ:

$$\mathrm{Adv}^{\texttt{SIM-EXT}}_{\Pi_{FS},\mathcal{R}}(S, \mathcal{E}, P^*, \lambda) := \left| \Pr[\texttt{SIM-EXT}^{S,P^*}(\lambda)] - \Pr[\texttt{SIM-EXT}_0^{\mathcal{E},S,P^*}(\lambda)] \right|.$$

Properties of SE

Π_{FS} is $\texttt{SIM-EXT}$ in the ROM if:

- It is adaptively knowledge sound.
- It is perfect k-ZK, meaning that there exists a simulator that perfectly simulates honest proofs, but only programs the RO when generating the k-th challenge.
- It is k-UR for the same round k, meaning no adversary can produce two accepting proofs that are identical up to the k-th round, even if it can program that round's challenge.

Preliminaries of SE

Let $\Pi = (\texttt{Setup}, P, V)$ be a $(2r+1)$-message public-coin interactive argument

- k-Zero-Knowledge: Let $\texttt{HVZK}$ simulator S, and $k \in [1, r]$. Let Π_{FS} be associated FS-transformed $\texttt{NIZK}$. Π_{FS} satisfies (perfect) k-zero-knowledge (k-ZK) if $\exists$ a zero-knowledge simulator $S_{FS,k}$ that only needs to program the random oracle in round k, and whose output is identically distributed to that of honestly generated proofs.
- k-Unique Response: Let Π_{FS} associated FS-transformed $\texttt{NARG}$ and $k \in [0, r]$. We say Π_{FS} satisfies k-unique response (k-UR) if $\forall$ $\texttt{PPT}$ adversaries A, the following probability is negligible in λ:

  $$\mathrm{Adv}^{k-UR}_{\Pi_{FS}}(A) := \Pr[k - UR_{\Pi_{FS}}(\lambda)].$$

  When $k = 0$, we say that Π_{FS} has (computationally) unique proofs.

```
**Game** $\mathsf{SIM\text{-}EXT}^{\mathcal{S},\mathcal{P}^*}_{0,\Pi_{\mathsf{FS}}}(\lambda)$
$\mathsf{pp} \leftarrow \mathsf{Setup}(\mathsf{pp}_{\mathcal{G}})$
$(x,\pi) \leftarrow (\mathcal{P}^*)^{\mathsf{H},\mathcal{S}}(\mathsf{pp})$
$b \leftarrow \mathcal{V}^{\mathsf{H}'}_{\mathsf{FS}}(\mathsf{pp},x,\pi)$
**return** $b \wedge (x,\pi) \notin \mathcal{Q}_{\mathsf{Sim}}$
```

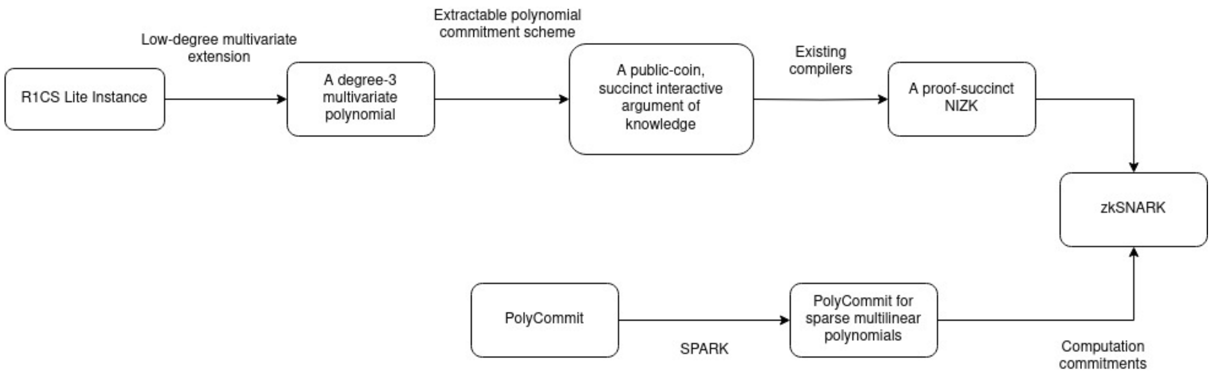
```
**Game** $\mathsf{SIM\text{-}EXT}^{\mathcal{E},\mathcal{S},\mathcal{P}^*}_{1,\Pi_{\mathsf{FS}},\mathcal{R}}(\lambda)$
$\mathsf{pp} \leftarrow \mathsf{Setup}(\mathsf{pp}_{\mathcal{G}})$
$(x,\pi) \leftarrow (\mathcal{P}^*)^{\mathsf{H},\mathcal{S}}(\mathsf{pp})$
$b \leftarrow \mathcal{V}^{\mathsf{H}'}_{\mathsf{FS}}(\mathsf{pp},x,\pi)$
$w \leftarrow \mathcal{E}^{\mathcal{P}^*}(\mathsf{pp},x,\pi)$
**return** $b \wedge (x,\pi) \notin \mathcal{Q}_{\mathsf{Sim}} \wedge (\mathsf{pp},x,w) \in \mathcal{R}$
```

Figure 3.1: Simulation Extractability Games

```
**Game** $k\text{-}\mathsf{UR}^{\mathcal{A}}_{\Pi_{\mathsf{FS}}}(\lambda)$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda,\mathsf{pp}_{\mathcal{G}})$
$(x,\pi,\pi',c) \leftarrow \mathcal{A}^{\mathsf{H}}(\mathsf{pp})$
$b \leftarrow \mathcal{V}^{\mathsf{H}[(\mathsf{pp},x,\pi|_k)\mapsto c]}_{\mathsf{FS}}(\mathsf{pp},x,\pi) = 1$
$b' \leftarrow \mathcal{V}^{\mathsf{H}[(\mathsf{pp},x,\pi'|_k)\mapsto c]}_{\mathsf{FS}}(\mathsf{pp},x,\pi') = 1$
**return** $b \wedge b' \wedge \pi \neq \pi' \wedge \pi|_k = \pi'|_k$
```

Figure 3.2: Security game for k-unique response. Here $H[(\mathrm{pp},x,\pi_k) \to c]$ denotes the random oracle where the input (pp,x,π_k) is programmed to output c.

Our Approach



R1CS Lite Instance
Low-degree multivariate extension
A degree-3 multivariate polynomial
Extractable polynomial commitment scheme
A public-coin, succinct interactive argument of knowledge
Existing compilers
A proof-succinct NIZK
zkSNARK
PolyCommit
SPARK
PolyCommit for sparse multilinear polynomials
Computation commitments

Figure 4.1: Viking Pipeline

We replace R1CS in the Spartan pipeline with R1CSLite. The simulation extractability of Spartan as proved in [DG23] requires that the k for k-ZK and k-UR is the same. They further show that Spartan satisfies this condition with $k = r - 1$ where $r = 7\mu + 11$, which does not change when we replace R1CS with R1CSLite.

Codebase

This necessitates the following changes to the pipeline:

- Created working instances in both the R1CS and R1CSLite constraint systems for the satisfiability of the polynomial relation $y = x^{16} + 1$, inspired by the blog written by Vitalik Buterin
- We started off by removing the C matrix across the huge codebase and then incrementally started fixing the approach
- We had to simulate the evaluation of the missing C matrix without actually storing the matrix anywhere. This was done for memory usage optimization (shown later in performance comparisions)
- The evaluation logic required quite a bit of effort from our side to get it right (and obviously gave rise to a lot of bugs which again took up lot of our time)
- The construction of the variable z, the assignment vector, was done with padding to the nearest power of two for efficiency purposes. We had to also keep track of the unpadded z vector due to the change in the condition for the new constraint system
- The original code uses batching for speedup and they used a batch size of 3 due to the necessity for the three polynomials A, B, and C. We had to modify it to 2, since R1CSLite needs only A and B matrices

- We found two issues in the original code - The first one was a wrong formulation of the example which was not created according to the actual R1CS definition. The second one was a trivial assert statement, which was comparing the equality of the same variable instead of another one. We have raised issues for both of these bugs in the original repo as well - [1], [2]

Sumcheck

For a given R1CS instance $\mathbb{x} = (\mathbb{F}, A, B, C, io, m, n)$, let $s = \lceil \log m \rceil$. Thus, we can view matrices $A, B, C \in \mathbb{F}^{m \times m}$ as functions with the following signature: $\{0,1\}^s \times \{0,1\}^s \to \mathbb{F}$. Specifically, any entry in them can be accessed with a $2s$-bit identifier (or two s-bit identifiers). Furthermore, given a purported witness w to $\ll$, let $Z = (io, 1, w)$. It is natural to interpret Z as a function with the following signature: $\{0,1\}^s \to \mathbb{F}$, so any element of Z can be accessed with an s-bit identifier.

We now describe a function $F_{io}(\cdot)$ that can be used to encode w such that $F_{io}(\cdot)$ exhibits a desirable behavior if and only if $\text{Sat}_{\text{R1CS}}(\mathbf{x}, w) = 1$.
Proof. This follows from the definition of $\text{Sat}_{\text{R1CS}}(\mathbf{x}, w)$ (Section 2.1) and of $Z(\cdot)$. Unfortunately $F_{io}(\cdot)$ is a function, not a polynomial, so it cannot be directly used in the sum-check protocol. But, consider its polynomial extension $\widetilde{F}_{io} : \mathbb{F}^s \to \mathbb{F}$.

$$\widetilde{F}_{io}(x) = \left(\sum_{y \in \{0,1\}^s} \widetilde{A}(x,y) \cdot \widetilde{Z}(y)\right) \cdot \left(\sum_{y \in \{0,1\}^s} \widetilde{B}(x,y) \cdot \widetilde{Z}(y)\right) - \sum_{y \in \{0,1\}^s} \widetilde{C}(x,y) \cdot \widetilde{Z}(y)$$

$$\Sigma_{r \in \{0,1\}^s} \tilde{F}(r) = 0$$

Unfortunately, this is not enough to conclude that $\tilde{F}(r) = 0$ for all r, because there may be some non-trivial cancellation in the sum from the higher degree terms.
Consider:

$$Q_{io}(t) = \sum_{x \in \{0,1\}^s} \widetilde{F}_{io}(x) \cdot \tilde{\text{eq}}(t, x)$$

where $\tilde{\text{eq}}\ (t, x) = \prod_{i=1}^{s} (t_i \cdot x_i + (1 - t_i) \cdot (1 - x_i))$.

Let $\widetilde{F}_{io}(r_x) = \bar{A}(r_x) \cdot \bar{B}(r_x) - \bar{C}(r_x)$, where

$$\bar{A}(r_x) = \sum_{y \in \{0,1\}^s} \widetilde{A}(r_x, y) \cdot \widetilde{Z}(y)$$
$$\bar{B}(r_x) = \sum_{y \in \{0,1\}^s} \widetilde{B}(r_x, y) \cdot \widetilde{Z}(y)$$
$$\bar{C}(r_x) = \sum_{y \in \{0,1\}^s} \widetilde{C}(r_x, y) \cdot \widetilde{Z}(y)$$

This observation opens up the following solution: the prover can make three separate claims to $\mathcal{V}$, say that $\bar{A}(r_x) = v_A, \bar{B}(r_x) = v_B$, and $\bar{C}(r_x) = v_C$. Then, $\mathcal{V}$ can evaluate:

$$\mathcal{G}_{io,\tau}(r_x) = (v_A \cdot v_B - v_C) \cdot\ \tilde{\text{eq}}\ (r_x, \tau),$$

which in turn enables $\mathcal{V}$ to verify $\mathcal{G}_{io,\tau}(r_x) \overset{?}{=} e_x$. Of course, $\mathcal{V}$ must still verify three new claims from $\mathcal{P}$: $\bar{A}(r_x) \overset{?}{=} v_A, \bar{B}(r_x) \overset{?}{=} v_B$, and $\bar{C}(r_x) \overset{?}{=} v_C$. To do so, $\mathcal{V}$ and $\mathcal{P}$ can run three independent instances of the sum-check protocol to verify these claims.

Code Analysis

GitHub Repository

Code can be found here

Table 5.1: Execution times for Spartan and Viking NIZK Proofs and Verifications (Constraint Size: 2^{10} to 2^{14})

Constraint Size →	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}
Spartan Proof	21.946ms	32.516ms	50.715ms	84.063ms	149.97ms
Viking Proof	21.395ms	32.037ms	49.684ms	83.324ms	149.07ms
Spartan Verify	7.1222ms	8.6966ms	10.190ms	12.813ms	16.113ms
Viking Verify	7.1775ms	8.2115ms	9.6508ms	12.009ms	15.825ms

Table 5.2: Execution times for Spartan and Viking NIZK Proofs and Verifications (Constraint Size: 2^{15} to 2^{20})

Constraint Size →	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}
Spartan Proof	268.65ms	509.46ms	888.15ms	1.7507s	3.1406s	6.2433s
Viking Proof	266.77ms	506.37ms	889.03ms	1.7441s	3.1151s	6.3057s
Spartan Verify	24.390ms	38.181ms	62.904ms	112.90ms	208.78ms	400.22ms
Viking Verify	23.215ms	36.762ms	62.260ms	113.47ms	209.47ms	408.92ms

Issues

During our study of the Spartan Code, we identified several issues that we have brought to the attention of the authors on Github. Some of them are as follows:

Issue: R1CS Instance Discrepancy

The first issue concerns the R1CS instance, which does not align with the defined standards. The R1CS instance example utilized in cubic.rs does not adhere to the R1CS instance criteria outlined in Theorem 19 of [CFF+21].

Theorem 19. Let $C : \mathbb{F}^{\ell_{\text{in}}} \times \mathbb{F}^{\ell_{\text{wit}}} \to \mathbb{F}^{\ell_{\text{out}}}$ be an arithmetic circuit with N multiplication gates. Then there exists an R1CS $L, R, O \in \mathbb{F}^{(N+\ell_{\text{out}})\times n}$ with $n = \ell_{\text{in}} + \ell_{\text{out}} + N + 1$ such that for any x, $\exists w : C(x, w) = y$ if and only if $\exists c$ that makes $(1, x, y)$ accepted by (L, R, O).

Although still a valid constraint system, it is less efficient. We aim for the total constraints to equal the number of multiplication gates plus one. The final constraint should manage all additions, hence the last two constraints can be merged into a single addition constraint.

Issue: Potential Mistake in Assert Statement

Another issue arises from a possible error in the assert statement found in src/sparse_mlpoly.rs at line 1265. The code snippet is as follows:

```
assert_eq!(eval_dotp_left.len(), eval_dotp_left.len());
```

It appears there might be a mistake in this statement. Perhaps the authors intended:

```
assert_eq!(eval_dotp_left.len(), eval_dotp_right.len());
```

Links

To the Issues - R1CS Issue, Assertion Issue

Conclusion

We have presented Viking, a non-malleable zkSNARK without trusted setup using the R1CSLite constraint system that operates only on the ROM and DLP assumptions through an extension of the work in Spartan [Set20]. Viking offers shortened proofs with slightly shorter to similar proving and verification times.

Future Work

Parallelization is an efficient way to increase the proving process's efficiency. The computational workload can be split over several cores or processors to greatly speed up the proving process. Several strategies can be used to accomplish this parallelization, such as dividing the activities involved in proof production into separate, concurrently processing parts. Parallelization not only makes it possible to lower the total time needed for proving, but it also improves the exploitation of computational resources, increasing the process's scalability and adaptability to various hardware configurations. A promising result is [Jia] who parallelize the original Spartan[Set20] paper.

In the field of post-quantum cryptographic protocols, the creation of simulation-extractable zk-SNARKs that do not rely on the difficulty of the discrete logarithm problem (DLP) is an urgent requirement. Such developments are critical in protecting the security and privacy of digital transactions and communications in the face of quantum computing threats since standard assumptions such as the hardness of integer factoring and DLP do not hold against a quantum adversary.

Challenges Faced

- Lots of cryptographic primitives and methods used to make zkSNARKs. Can be tough to understand.
- Large codebase (8000 lines) in a language which we are not comfortable with (Rust)
- Complex implementation which was hard to adapt to R1CSLite, especially since replacing R1CS with R1CSLite involved modifying a large number of functions in the codebase

Work Division

- Gnana Prakash Punnavajhala - Code(Major), Report, Slides, Paper Review
- Ishwar B Balappanawar - Code, Report, Slides, Building R1CSLite, Paper Review
- Shrikara A - Simulation Extractability Proof, Primitives, Slides, Paper Review
- Vansh Pravin Marda - Code Documentation, Paper Review, Report
- Yash Prashant Adivarekar - Code Documentation, Paper Review, Report

Bibliography

[ALM$^+$98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM (JACM)*, 45(3):501–555, 1998.

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of np. *Journal of the ACM (JACM)*, 45(1):70–122, 1998.

[BBCG$^+$19] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. Zero-knowledge proofs on secret-shared data via fully linear pcps. In *Annual International Cryptology Conference*, pages 67–97. Springer, 2019.

[BCCT12] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 326–349, 2012.

[BCG$^+$20] Sean Bowe, Alessandro Chiesa, Matthew Green, Ian Miers, Pratyush Mishra, and Howard Wu. Zexe: Enabling decentralized private computation. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 947–964. IEEE, 2020.

[BFLS91] László Babai, Lance Fortnow, Leonid A Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 21–32, 1991.

[BGH19] Sean Bowe, Jack Grigg, and Daira Hopwood. Recursive proof composition without a trusted setup. Cryptology ePrint Archive, Paper 2019/1021, 2019. https://eprint.iacr.org/2019/1021.

[BPW12] David Bernhard, Olivier Pereira, and Bogdan Warinschi. *How not to Prove Yourself: Pitfalls of the Fiat-Shamir heuristic and applications to helios.* 1 2012.

[BSCG$^+$13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. Snarks for c: Verifying program executions succinctly and in zero knowledge. In *Annual cryptology conference*, pages 90–108. Springer, 2013.

[BSCTV14] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct {Non-Interactive} zero knowledge for a von neumann architecture. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 781–796, 2014.

[CFF$^+$21] Matteo Campanelli, Antonio Faonio, Dario Fiore, Anaïs Querol, and Hadrián Rodríguez. Lunar: a toolbox for more efficient universal and updatable zksnarks and commit-and-prove extensions. In *Advances in Cryptology–ASIACRYPT 2021: 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6–10, 2021, Proceedings, Part III 27*, pages 3–33. Springer, 2021.

[DG23] Quang Dao and Paul Grubbs. Spartan and bulletproofs are simulation-extractable (for free!). In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 531–562. Springer, 2023.

[GKK$^+$22] Chaya Ganesh, Hamidreza Khoshakhlagh, Markulf Kohlweiss, Anca Nitulescu, and Michał Zajac. What makes fiat–shamir zksnarks (updatable srs) simulation extractable? In *International Conference on Security and Cryptography for Networks*, pages 735–760. Springer, 2022.

[GOP$^+$22] Chaya Ganesh, Claudio Orlandi, Mahak Pancholi, Akira Takahashi, and Daniel Tschudi. Fiat–shamir bulletproofs are non-malleable (in the algebraic group model). In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 397–426. Springer, 2022.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. Cryptology ePrint Archive, Paper 2016/260, 2016. https://eprint.iacr.org/2016/260.

[HLPT20] Thomas Haines, Sarah Jamie Lewis, Olivier Pereira, and Vanessa Teague. How not to prove your election outcome. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 644–660. IEEE, 2020.

[Jia] Jiangkm. Github - jiangkm3/spartan$_parallel : Spartan : High - speedzksnarkswithouttrustedsetup$.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 723–732, 1992.

[LSZ22] Helger Lipmaa, Janno Siim, and Michał Zajac. Counting vampires: from univariate sumcheck to updatable zk-snark. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 249–278. Springer, 2022.

[SCG$^+$14] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE symposium on security and privacy*, pages 459–474. IEEE, 2014.

[Set20] Srinath Setty. Spartan: Efficient and general-purpose zksnarks without trusted setup. In *Annual International Cryptology Conference*, pages 704–737. Springer, 2020.

[Z.C] Z.Cash. Zcash: Privacyx2d;protecting digital currency.