RedShift: Transparent SNARKs from List Polynomial Commitment IOPs

Assimakis Kattis
New York University
kattis@cs.nyu.edu

Konstantin Panarin
Matter Labs
Higher School of Economics
kp@matterlabs.dev

Alexander Vlasov
Matter Labs
av@matterlabs.dev

***Abstract*—We introduce an efficient transformation from univariate polynomial commitment based zk-SNARKs to their fully transparent counterparts. The transformation is achieved with the help of a new IOP primitive which we call a list polynomial commitment. This primitive is applicable for preprocessing zk-SNARKs over both prime and binary fields. We present the primitive itself along with a soundness analysis of the transformation and instantiate it with an existing universal proof system. We also present benchmarks for a proof of concept implementation alongside a comparison with a non-transparent alternative based on Kate commitments. Our results show competitive efficiency both in terms of proof size and generation times at large security levels.**

***Index Terms*—polynomial commitments, zero-knowledge proofs, proximity testing, verifiable computation**

I. Introduction

Zero-knowledge proofs [1] have recently received increased amounts of attention for providing efficient verification while maintaining small proof sizes, even in the case of complex predicates. Initially limited to theoretical considerations, such proof systems have lately come to encompass the underlying technology in a wide variety of practical and industrial applications with delicate trade-offs between privacy and system security [2] [3] [4]. In this work, we are interested in applications for which there is limited space availability in the underlying system, and thus for which minimal proof size is an important property. Moreover, we ideally want to focus on applications for which there exist no trusted parties at any point of the computation, and thus hope to achieve proof size minimization *without* compromising the trust model of the system.

The trade-off above is most closely associated with applications of zero-knowledge proofs to cryptocurrency systems, such as Ethereum [5] or ZCash [6], in which participants have to verify state (or transaction) validity to ensure system soundness but for which there is limited space available in which to do so. Bridging the gap between these two requirements will allow for not only efficient but also trustless verification of state transition in such systems. This has the potential for scaling improvements, such as increased transaction throughput or better privacy guarantees.

The most widely used proof systems for such an application are preprocessing Succinct Non-interactive ARguments of Knowledge (zk-SNARKs) [7] [8] [9], for which proof size and verification time are polylogarithmic in the size of the circuit being verified. 'Pre-processing' here denotes that such systems rely on a one-time (often expensive) setup procedure to produce a proving/verification key-pair (pk, vk) (known as a Structured Reference String or SRS) that is used in all subsequent computation. The most efficient such construction is due to Groth [7] and achieves constant proof size consisting of 3 group elements, with state-of-the-art proving time. However, this construction (along with most in the literature, see [10], [11], [12]) relies on a *trusted setup*, or a trusted third-party actor to generate certain parameters (known as the 'toxic waste') that should be destroyed in order for the system to retain its security guarantees.

Such a security lapse would be grave for all aforementioned applications. For example, in a cryptocurrency system such as ZCash an adversary possessing such waste would be able to spend non-existent tokens without being found. An adopted approach to mitigating this issue involves Multi-Party Computation, in which a single participant needs to destroy their parameters for security to hold [13]. However, scaling such an approach to many participants comes with its own challenges, and can never reach the completely trustless threat model desired by such systems.

The trust issue inherent in the above approach stems from the requirements for the generation of the SRS of the proof at the preprocessing stage. This is done once at the beginning of the protocol, encoding information that is used in the subsequent proof generation of any input arguments. More specifically, in most pairing-based SNARKs (such as [14]) the trusted part of SRS generation stems from the usage of a polynomial commitment scheme that needs to sample (secret) randomness in order to provide commitments to some low-degree polynomial that in turn encodes the circuit in question. That information is then used by the prover to efficiently convince the verifier that a given value is indeed the evaluation of this polynomial, thus proving knowledge of the statement. Such systems use the polynomial commitment scheme of [15], from which the above trust model is derived. This will be further discussed in the following sections.

In attempting to retain a trustless (or 'transparent') threat model, the main design challenge lies in the efficiency of the underlying protocol. Various threads of work in this domain have achieved different efficiency trade-offs. The work of [16] produces proofs with size scaling as $O(d \log T)$, while the proofs in [17] scale with $O(d \log G)$ where T, d and G the size, depth and width of the circuit respectively. Succinct Transparent ARguments of Knowledge (zk-STARKs) [18] achieve $O(\log^2 T)$ proof sizes for uniform (layered) circuits. However, in the context of *universal* SNARKs (arbitrary circuits), existing proof systems suffer from performance overheads with respect to pre-processing SNARK constructions such as [7]. Some also require non-trivial circuit designs, similar to what is described in [19]. Nevertheless, we should note that for the class of problems that can be efficiently expressed as layered circuits, these proof systems may be more optimal than universal ones. Since we are also interested in verifier succinctness, transparent approaches such as [20] do not suffice here due to the linear dependence between verification time and predicate size.

Below we informally describe the properties that an 'ideal' proof system should possess for satisfiability of a given circuit C, where $|C|$ denotes its size. The first three properties define what is known as a 'fully succinct' zk-SNARK construction:

- *Verifier Succinctness:* Verification time is polylogarithmic in $|C|$.
- *Prover Efficiency:* Proving time is quasi-linear in $|C|$.
- *Proof Succinctness:* Proof size is poly-logarithmic in $|C|$.
- *Transparent:* No trust assumptions are required for security to hold.
- *Plausibly Quantum Resistant:* The system is not be based on assumptions known to be false in a quantum setting.

A. Prior & Concurrent Work

1) Transparent zk-SNARK Constructions: A new approach to the above problem relies on creating a 'universal' SRS at the preprocessing phase, which can then be used in tandem with *any* possible predicate (or circuit). This has been the focus of many recent contributions (see [14], [21], [22]) and most recently [23] that are also fully succinct zk-SNARKs in the above sense. The approach in such schemes relies on two main ingredients: (1) encoding the circuit satisfaction problem of the predicate in question as a property of some (low-degree) polynomial f, and then (2) committing to f using a polynomial commitment scheme. In all the above approaches, the polynomial commitment scheme in [15] is used due to its constant size complexity and efficient implementation. However, this is the only part in the protocol that introduces the trusted setup, as the setup phase in the scheme requires a trusted actor to create (and then destroy) a secret value that is only used in generating commitments.

2) Polynomial Commitment Schemes: At a high level, polynomial commitment schemes allow for the efficient verification of the evaluations of f at an arbitrary point in its domain. Given the above contributions, it is immediate that a polynomial commitment scheme that is both transparent and efficiently computable would yield transparent SNARK constructions that have the potential to satisfy all of the requirements of a fully succinct, transparent and plausibly quantum resistant zk-SNARK. Since the introduction of polynomial commitment schemes in [15], the first transparent such scheme was introduced in [17] for multivariate polynomials, with $O(\sqrt{d})$ commitment size and verification complexity. Subsequent work in [24] introduces a scheme with $O(\mu \log d)$ size and verification complexity, where μ the number of variables of the polynomial in question and d the polynomial's degree. Although the asymptotics of the approach in [24] suffice for the above motivation, the practical implementation of their system relies on cryptographic operations that are substantially more resource-heavy than previous approaches. This stems from the reliance of their system's security on class groups of unknown order. Although the proof sizes achieved are sufficiently succinct, this dependence could make practical deployment difficult at reasonable security levels when proof generation time needs to also be substantially efficient. Moreover, the assumptions on which their construction rests are known to not be quantum-resistant.

3) Preprocessing zk-SNARK Compilation Frameworks: Recent work has also explored general frameworks for converting Interactive Oracle Proofs (IOPs) [25] into preprocessing SNARKs. This approach was introduced in [26], with an equivalent formalization appearing in [24]. At a high level, both of these contributions formalize the idea that preprocessing zk-SNARKs can be constructed from IOPs through oracle access to a low-degree polynomial.

B. Our Contributions

In this work, we provide the following contributions:

1) List Polynomial Commitments: The works of [27] and [28] introducing the Fast Reed Solomon IOP of Proximity (FRI IOPP) implicitly define a transparent polynomial commitment scheme, which provides commitments of size $O(\log^2 d)$ for polynomials of degree d. However, the soundness error on such schemes implies that high-security deployments still suffer from large proof sizes. In this work, we leverage the proof system threat model to 'relax' the requirements on transparent polynomial commitment schemes while still retaining all necessary security properties for compilation into zk-SNARKs. We introduce a new cryptographic primitive for fast verification of polynomial evaluations we call a *list commitment scheme*. At a high level, this scheme provides the necessary security guarantees inherent in polynomial commitment schemes that are required for polynomial-based proof systems such as [23] and [14]. In the language of IOP formalization, this primitive can be thought of as an alternative compiler for public-coin IOP protocols.

2) Compilation of IOPs with List Commitments: The above contribution implicitly provides a general framework that demonstrates how the polynomial list commitment can be used to compile any polynomial IOP into a preprocessing zk-SNARK. As previously mentioned, this follows the approach in [26] and [24] with the main difference being that we do not require a polynomial commitment scheme in their (more restrictive) sense.

3) RedShift: A transparent zk-SNARK Construction: We demonstrate the security and practicality of this approach by compiling [23] using the framework above. By fitting an implementation of the list commitment scheme on [23] with suitable adaptations and optimizations, we remove all trusted computation while retaining efficiency in both proof size and generation time. We call this new proof system REDSHIFT, and provide:

1) formal proofs of correctness and security for the proposed system, demonstrating that this protocol is a zk-SNARK,
2) a proof-of-concept implementation, along with benchmarks establishing feasibility.

Benchmarks and proof sizes are provided in Section X. Overall, REDSHIFT is an efficient instantiation of a post-quantum transparent preprocessing zk-SNARK suitable for practical deployment at high security levels. We also demonstrate how varying the list size allows for proof size improvements. This, however, comes at the cost of larger proof generation times and RAM consumption.

II. OVERVIEW

The proof system in [23] (known as PLONK) is based on polynomial commitment schemes. The role of polynomial commitments in PLONK is the following: a secret prover's witness is encoded as a set of univariate polynomials, while the verifier wishes to ensure that such an encoding satisfies some polynomial relations. The prover commits to her secret witness and later the verifier queries the values of witness polynomials at a set of randomly selected points, checking if all relations are indeed satisfied at those points. As the points were sampled uniformly randomly, it is highly likely that the given polynomial relations are identically satisfied.

The state-of-the-art polynomial commitment scheme used in the construction of ZK-protocols is the KATE commitment ([15]), which is based on pairings of points of elliptic curves. The security of this scheme reduces to the Discrete Log assumption while in the case of a perfectly hiding commitment a strengthened version known as the t-Strong Diffie Hellman assumption is required. The main drawback of [15] lies in the requirement that some secret value needs to be sampled as part of the parameter generation process. For security to hold, this value should never be revealed to the prover nor the verifier. Such a requirement is very strong, as it means that every proof system using KATE commitment (and PLONK is among them) as a sub-protocol will inevitably require a 'trusted' setup ceremony. We denote proof systems without this requirement as *transparent*.

In fact, the only reason PLONK requires a trusted setup ceremony is due to the KATE sub-protocol. This means that once we replace such a polynomial commitment scheme by any transparent equivalent, we will not have to conduct a trusted setup ceremony. Given this, this work aims to find a suitable replacement for KATE, turning PLONK (and similar systems) into a zero-knowledge, succinct transparent argument of knowledge, i.e. a zk-STARK.

To do so, we utilize the FRI protocol, which is a key component of STARK [18] and AURORA [29]. FRI is focused on solving the following *proximity* problem: The verifier is given oracle access to an evaluation of some function f on a fixed domain $D \subset \mathbb{F}$. The prover wants to convince the verifier that this function f is close in some metric to a polynomial of some predefined degree d. The simplest solution would be for verifier to query all the values of f on the domain and then compute interpolation polynomial herself and check that its degree is indeed bounded above by d. However this straightforward approach requires large communication complexity (as the verifier queries *all* the values) and carries a large computational burden. FRI solves this problem by requiring only a polylogarithmic number of queries in d.

The naive approach to build a (transparent) polynomial commitment scheme on the top of FRI *would be* the following:

1) The prover commits to f providing an oracle to all evaluations of f on some predefined domain D.
2) The prover and verifier engage in the FRI protocol for a function f w.r.t to some degree d. If the prover passes the check the verifier is then convinced that the function f is actually close to a polynomial of degree less than d
3) The verifier wants to retrieve the value of f at point $i \notin D$. The prover sends the corresponding opening $z = f(i)$ and prover and verifier then conduct another instance of FRI, this time w.r.t to a quotient function $q(x) = \frac{f(x)-z}{x-i}$ and degree $< d-1$. Note that the verifier has oracle access to $q(x)$ via oracle access to f and known i and z. If the prover passes the last instance of FRI then $q(x)$ is in fact a polynomial function of degree $< d-1$ (*a priori* we only know it is a rational function) which implies $f(i) = z$. This follows from Bezout's theorem, stating that $h(x)$ has value y at point t iff $h(x) - y$ is divisible by $x - t$ in the ring $\mathbb{F}[x]$.

In reality this simplified protocol doesn't suffice. There are several reasons for that, among which are the following:

1) FRI has a sensitivity bound - it is incapable of distinguishing between precise polynomials and functions which are not polynomials but sufficiently close to them in some predefined metric (which in our case is the relative Hamming distance).
2) For implementation coherency, we want the same domain for both FRI instances. However, FRI has an interdependence between the degree d and the size of the domain $|D|$ measured in terms of its rate $\rho = \frac{|D|}{d}$. The divide-and-conquer nature of FRI requires the rate to be "2-adic", that is of the form 2^{-R} for some R in $\mathbb{N}$. However, this property cannot hold simultaneously for two adjacent degrees d and $d-1$ without extra protocol modification.

The first mentioned problem means that the scheme needs to correctly process the case when the function is not a polynomial, but close to one - a property not naturally supported by existing commitment schemes. Even more so, allowing the oracle f to not be strictly polynomial and to take as the prover's commitment the polynomial f' lying in a small δ-ball around f (where δ is taken according to sensitivity of FRI) then *a priori* we cannot guarantee this polynomial f' is unique in the chosen neighborhood of f. We call the set of polynomials $\{f'_1, f'_2, \dots, f'_n\}$ lying in the δ-neighborhood of

f to be the *δ-list* of f and denote it by $L_\delta = L_\delta(f)$. For small values of δ the list L_δ contains precisely one polynomial: in this case we say that δ lies in the *unique-decoding radius*. The problem is that such values of δ require larger proof sizes to achieve adequate soundness guarantees. Thus, we would need to increase δ to reduce proof sizes, which in turn would lead to the size L_δ be greater than 1.

To solve this, we consider a *relaxed* treatment of commitment schemes, where the commitment opens to a polynomial in the δ-list L_δ. When the prover is asked for an evaluation at point i, they respond with some value $f'(i)$, where $f' \in L_\delta$. In subsequent sections we show that this scheme is sufficient for the construction of a transparent PLONK instance.

During the execution of PLONK both the prover and verifier need to evaluate a set of 'constraint' (or setup) polynomials $c_j(x)$ which encode the constraint system itself and which are known by both parties from the very beginning. In order to achieve succinctness, the verifier never calculates the value $c_j(i)$ at point i by herself. To resolve this *polynomial evaluation problem*, PLONK instead relies on the Kate scheme: the prover and verifier run the commitment protocol with the commitment to c and value i as inputs. By the security of Kate, the verifier is convinced that the prover actually sends the evaluation $c(i)$ of the polynomial $c(x)$ in question. Since our relaxation commits to a whole neighborhood $L_\delta(c)$ of $c(x)$ instead of only $c(x)$ itself, we lose this uniqueness property. This means we can't use such a 'relaxed' scheme as is. However, we show that with minor changes, our list commitment can be turned into a polynomial evaluation scheme. This transformation constitutes the second key sub-protocol of the paper.

With the list commitment and evaluation schemes, we can modify PLONK to achieve full transparency. We call the modified version RedShift and prove its correctness in the IOP model. A large portion of our approach remains the same as in [23]: our modification doesn't touch the completeness property of the system. However, the FRI-based protocol doesn't possess the hiding capabilities of Kate. This means that we need to take additional measures to achieve zero-knowledge for our system. We also need to change the security model - the original PLONK protocol was proven secure in the algebraic group model. Our approach is highly dependent on FRI - an IOPP protocol. This means that we need to conduct our security analysis in the IOP-model as well. Changing the threat model affects the soundness proof as well as the proof of knowledge approaches.

A. Roadmap

- In Section III, we provide general definitions, notations and terms used throughout the paper. We also explain the properties of the threat model used throughout this paper.
- Section IV provides a detailed overview of the FRI algorithm. We do not consider FRI as a black-box protocol for solving the proximity problem, as the knowledge of FRI's inner functionality will be required in the proof of zero-knowledge.
- In Section V we describe the key component of RedShift - the 'list polynomial commitment' scheme, and prove that it meets all the requirements.
- In Section VI we present the evaluation scheme that is used for 'constraint' polynomials.
- At this point we have all the components required for the transformation of PLONK into RedShift, which is conducted in Section VII.
- In Section VIII we will discuss various optimizations and variants for the final protocol.
- In section IX we discuss the particular choice of setup parameters leading to the most effective instantiation of RedShift.
- In Section X we measure the running time of our protocol as compared to the original PLONK construction over various pairing-friendly curves.

III. Definitions

In this section, we lay out the building blocks that are necessary to describe our constructions.

A. Notation

Through this paper we use the following notations:

- $\mathbb{F}_q$ is a prime field with modulus q
- $D \subset \mathbb{F}$ is an evaluation domain for Reed Solomon code words
- $f|_D$ is a restriction of function f to domain D
- For function pair f, g, the relative Hamming distance with respect to some domain D is given by:

$$\Delta(f,g) = \frac{\#\{x \in D : f(x) \neq g(x)\}}{|D|}.$$

B. Preliminaries on Reed-Solomon codes

Most of the information covered in this section can be found in any standard textbook on algebraic codes(e.g. [30]).

Definition 1 (Reed-Solomon Codes). *For some subset D of a given field $\mathbb{F}$ and a rate parameter $\rho \in (0,1]$, we denote by $\mathsf{RS}[\mathbb{F}, D, \rho]$ the set of all functions $f : D \to \mathbb{F}$ that are polynomials of degree $d < \rho|D|$. A binary additive RS code family is a code family $\mathsf{RS}[\mathbb{F}, D, \rho]$ for which $\mathbb{F} = \mathbb{F}_{2^m}, m \in \mathbb{N}$. Moreover, the set D is required to be an additive coset which is an additive shift of some $\mathbb{F}_2$-linear space in $\mathbb{F}_{2^m}$. A prime field RS code family is a code family $\mathsf{RS}[\mathbb{F}, D, \rho]$ for which $\mathbb{F} = \mathbb{F}_q$, for prime q. In this case D is a multiplicative subgroup of $\mathbb{F}_q^*$.*

Definition 2 (List Decoding). *Let $V = \mathsf{RS}[\mathbb{F}, D, \rho] \subset \mathbb{F}^D$ be an RS code family. Set a distance parameter $\delta \in [0,1]$. For $u \in \mathbb{F}^D$, we define $L(u, V, \delta)$ to be the set of elements in V that are at most δ-far from u in relative Hamming distance. The code V is said to be (δ, N)-list-decodable if $|L(u, V, \delta)| \leq N$ for all $u \in \mathbb{F}_q^D$. Let $L_\delta = L(\mathbb{F}, D, d, \delta)$ be the maximum size of $L(u, V, \delta)$ taken over all $u \in \mathbb{F}^D$ for $V = RS[\mathbb{F}, D, \rho = d/|D|]$.*

Theorem 1 (List Decoding Johnson bound [28])**.** *Fix $\rho \in (0, 1)$. Then for $\mathsf{RS}[\mathbb{F}, D, \rho]$ list size $|L|$ is*

$$J_{\rho,\epsilon} = max(1 - \sqrt{\rho} - \epsilon, \frac{1}{2\epsilon\sqrt{\rho}})$$

for every $\epsilon \in (0, 1 - \sqrt{\rho})$.

The natural question arising in this context is the following: for which distance parameters δ do we get unique decodability (i.e. $L_\delta \leq 1$)?

Definition 3. *We call δ_0 unique decoding radius if for all $\delta < \delta_0$ list size $L_\delta \leq 1$. We call such $\delta < \delta_0$ to lie in the unique decoding radius.*

The following theorem is a well-known fact on unique decodability for Reed-Solomon facts.

Theorem 2. $\delta_0 = \frac{1-\rho}{2}$ *is the unique decoding radius for $\mathsf{RS}[\mathbb{F}, D, \rho]$*

The decoding problem for Reed-Solomon code $V = \mathsf{RS}[\mathbb{F}, D, \rho]$ is the problem of finding a codeword $u \in V$ that is within a distance of δ (with respect to Hamming distance) from a "received" word $\in \mathbb{F}^D$. There are two famous polynomial-time solutions for decoding problem. The first is the classical *Berlekamp-Messy* [31] [32] algorithm which can be applied only in the unique decoding radius setting. It's extension for distance parameters outside the unique-decoding bound is *Guruswami-Sudan* [33] algorithm. As the result of the latter algorithm we will get *all* codewords lying in δ-ball of a "received" word.

Theorem 3 (Berlekamp-Messy algorithm)**.** *For every $\delta < \delta_0$ the decoding problem for $V = \mathsf{RS}[\mathbb{F}, D, \rho]$ can be solved in $O(|D|^3)$ steps.*

Theorem 4 (Guruswami-Sudan algorithm)**.** *For all $\delta \leq 1 - \sqrt{\rho}$ Reed-Solomon code $V = \mathsf{RS}[\mathbb{F}, D, \rho]$ can be list-decoded in time $O(|D|^{15})$. When $\delta < 1 - \sqrt{\rho}$ then the decoding time reduces to $O(|D|^3)$.*

C. Interactive Oracle Proofs and IOPs of Proximity

For user convenience we will start with a remainder of properties of ZK-schemes and of threat model we are going to use. Given a relation $\mathcal{R} \subseteq S \times T$, we denote by $\mathcal{L}(\mathcal{R}) \subseteq S$ the set of $s \in S$ such that there exists $t \in T$ with $(s, t) \in \mathcal{R}$; for $s \in S$, we denote by $\mathcal{R}|_s \subseteq T$ the set $\{t \in T : (s, t) \in \mathcal{R}\}$. For pairs $(\phi, w) \in \mathcal{R}$ we call ϕ the statement and w the witness.

The security analysis in this section will be conducted in *Interactive Oracle Proof* (IOP) model [25] which is a simultaneous generalization of Interactive Proofs and Probabilistically Checkable Proofs. The input of the verifier is $x \in S$, and the input of the prover is $(x, w) \in \mathcal{R}$ for some string $w \in T$. The number of interactive rounds, denoted $r(x)$, is called the round complexity of the system. During a single round the prover P sends a message π_i (which may depend on prior interaction) to which the verifier V is given oracle access, and the verifier responds with message m_i to the prover. The output of V after interacting with P is either *accept* or *reject* and we denote the result of this interaction by $\langle P(x, w) \leftrightarrow V(x)\rangle$ The proof length, denoted $l(x)$, is the sum of lengths of all messages sent by the prover. The query complexity of the protocol, denoted $q(x)$, is the total number of entries read by V across various prover oracles.

A cryptographically secure IOP protocol should have the following properties.

Perfect completeness: This says that, given any true statement, an honest prover should be able to convince an honest verifier. For all $(\phi, w) \in \mathcal{R}$:

$$Pr\left[\langle P(\phi, w) \leftrightarrow V(\phi)\rangle = acc \;|(\phi, w) \in \mathcal{R}\right] = 1$$

.

Soundness: This says that malicious prover has negligible chance to convince verifier in the wrong witness. For every instance $x \notin \mathcal{L}(\mathcal{R})$ and unbounded malicious prover P^*:

$$Pr\left[\langle P^*(\phi, w) \leftrightarrow V(\phi)\rangle = acc \;|(\phi, w) \notin \mathcal{R}\right] \leq \epsilon(x)$$

with $\epsilon(x) \to 0$ when $|x| \to \infty$.

Knowledge soundness: Strengthening the notion of soundness, we say the IOP has knowledge soundness if every prover who is capable of convincing the verifier that $x \in \mathcal{L}(\mathcal{R})$ actually knows some witness $w \in \mathcal{R}|_x$. In other words IOP is knowledge sound if for all adversaries $\mathcal{A}$ there exists a non-uniform polynomial time extractor $\mathcal{E}(\mathcal{A})$ which gets full access to the adversary's state, including any random coins and with high probability computes a witness whenever the adversary produces a valid argument. Formally written:

$$\Pr\left[\; w \notin \mathcal{R}|_x \;\middle|\; \begin{matrix} \langle \mathcal{A}(\phi) \leftrightarrow V(\phi)\rangle = \text{acc} \\ w \leftarrow \mathcal{E}(\mathcal{A}) \end{matrix} \;\right] \leq \epsilon(x)$$

where $\epsilon(x)$ has the same properties as above.

The definition of zero knowledge property that we use for IOPs first requires the notion of a view, which we take them from [25].

Definition 4. *Let A, B be algorithms and x, y strings. We denote by $View(B(y), A(x))$ the view (or transcript) of $A(x)$ in an interactive oracle protocol with $B(y)$, i.e., the random variable$(x, r, a_1, ..., a_n)$ where x is A's input, r is A's randomness, and $a_1, \cdots, a_n$ are the answers to A's queries into B's messages.*

Zero knowledge: $\langle P, V\rangle$ has the zero knowledge property if there exists a probabilistic polynomial time algorithm S (the simulator) such that, for every $(\phi, w) \in \mathcal{R}$ and unbounded distinguisher D the following probabilities are equal:

$$Pr[D(View(P(\phi, w), V(\phi))) = 1] = Pr[D(S(\phi)) = 1].$$

Remark: In subsequent sections we will use the words *view* and *transcript* interchangeably.

An important subclass of IOP protocols is given by the following definition:

IOPP. An *Interactive Oracle Proof of Proximity (IOPP)* is an r-round interactive IOP for the following problem: given a field $\mathbb{F}$, $d \in \mathbb{N}$, $\delta > 0$ and domain $D \subset \mathbb{F}$, the prover is provided with the representation of some function f and the verifier is given oracle access to its evaluation on domain D (i.e. an oracle $\hat{f}(x)$ to $f(x)|_D$). The prover then needs to convince the verifier that $f|_D$ is in fact evaluations of some degree d-polynomial on this domain, namely that $f \in C$, where $C = \mathsf{RS}[\mathbb{F}, D, \rho = d/|D|]$. An IOPP of proximity must have the following properties:

1) First message format: the first prover message, denoted f^0, is a purported codeword (evaluation of $f(x)$ on the domain D)
2) Completeness:
$$\Pr[\langle P \leftrightarrow V \rangle = accept \,|\Delta(f, C) = 0] = 1$$
3) Soundness error is a function $err(\delta)$ such that the following equation holds: For any P^*,
$$\Pr[\langle P^* \leftrightarrow V \rangle = accept \,|\Delta(f, C) > \delta] \leq err(\delta).$$

IV. FRI

As a particular instance of an IOPP protocol, we use FRI [27], [34] which is state-of-the-art to the best of our knowledge. Here we provide an overview of its properties. Fix RS code family $\mathsf{RS}[\mathbb{F}, D, \rho]$ for which $|D| = n = 2^k$ and rate $\rho = 2^{-R}$ ($k, R \in \mathbb{N}$). This implies that degree bound d is 2^{k-R}. Fix $r \in [1, \log d = k - R]$ (number of FRI inner rounds). With such a choice of parameters the FRI IOPP has the following properties:

1) **Prover Complexity:** $O(n)$ arithmetic operations over $\mathbb{F}$.
2) **Verifier Complexity:** $O(\log n)$ arithmetic operations over $\mathbb{F}$.
3) **Completeness:** If $f \in \mathsf{RS}[\mathbb{F}, D, \rho]$ and the prover is honest, then the verifier always accepts.
4) **Soundness:** For every $\epsilon \in (0, 1]$, let $J_\epsilon : [0, 1] \to [0, 1]$ be the function
$$J_\epsilon(x) = 1 - \sqrt{1 - x(1 - \epsilon)}.$$
Suppose that $\Delta(f, \mathsf{RS}) = \delta > 0$, then soundness error $\mathsf{err}(\delta)$ of FRI is bounded above by (again for any $\epsilon \in (0, 1]$):
$$\frac{2 \log |D|}{\epsilon^3 |\mathbb{F}|} + \Big(1 - min\Big\{\delta_0, J_\epsilon(J_\epsilon(1 - \rho))\Big\} + \epsilon \log |D|\Big)^l$$
where l is a repetition parameter.

Remark 1: l repetition parameter is a number of queries performed during the FRI proof verification.

Remark 2: Later this soundness bound was improved in [28] using sampling out of the domain techniques.

V. List Polynomial Commitment (LPC) Scheme

We are now ready to introduce the main ingredient underlying the transparency of our proving system, which we call a list commitment scheme. This cryptographic primitive most resembles a polynomial commitment scheme, with the main difference being that the commitment is to an ϵ-ball around some polynomial f (in some predefined metric Δ), rather than specifically to f itself. As we are working in the IOP model we prefer not to dissect the protocol as a tuple of (commit, open, verify) sub-protocols, as it is commonly done in commitment schemes' literature and simply describe our relaxed scheme as a subclass of IOP protocol with specific semantics and structure. As before, we denote $L_\delta(f)$ as the δ-list of $f(x)$ or the set of all $g(x) \in \mathsf{RS}[\mathbb{F}, D, \rho]$ such that $\Delta(f, g) < \delta$.

1) LPC protocol semantics.:

1) **Setup:** The prover and verifier agree on the following parameters: field $\mathbb{F}$, domain $D \subset \mathbb{F}$, $\delta > 0$ (error-bound), $d \in \mathbb{N}$ (bound for degree of polynomial) and k (the number of points to open).
2) **First message format:** The first prover message is an oracle c to the evaluation of $f(X)$ on the domain D. This is analogous to the Commit method in KATE notation.
3) **Second message format:** The verifier chooses and sends to the prover a set of k points: $\{i_j\}_{j=1}^k$.
4) **Third message format:** The prover responds with values $\{z_j\}_{j=1}^k$, which are the purported openings for the $\{i_j\}_{j=1}^k$.
5) **Subsequent interaction:** The prover and verifier engage in a sub-protocol $\mathsf{MultiEval}\langle P, V\rangle$ in which the prover aims to convince the verifier of the validity of statement $\mathcal{R}^\delta(\mathsf{pp} = \langle c, \{i_j\}_{j=1}^k, \{z_j\}_{j=1}^k, d\rangle)$, defined by:
$$\mathcal{R}^\epsilon(\mathsf{pp}) = \left(\begin{array}{c} \exists\, g(x) \in \mathbb{F}_{<d}[x],\ \Delta(f, g) < \epsilon \\ g(i_j) = z_j \ \forall\ j \in \{1, \ldots, k\} \end{array} \right)$$

2) LPC Instantiation: We provide an implementation of the $\mathsf{MultiEval}\langle P, V\rangle$ sub-protocol for the case when $d = 2^n + k, k << d$, (n is any integer $\in \mathbb{N}$) and D - any FRI-friendly domain $\subset \mathbb{F}$.

1) Both the prover and verifier compute an interpolation polynomial $U(x)$ of degree less than k such that $U(i_l) = z_l$ for $l \in [k]$. Note that since the verifier needs to evaluate $U(x)$ themselves, we require $k = O(\log d)$. This means verifier can construct and evaluate $U(x)$ efficiently without any help from the prover.
2) The prover and verifier engage in an IOPP (here instantiated by FRI) protocol w.r.t to the quotient function:
$$q(x) = \frac{f(x) - U(x)}{\prod_{l=1}^k (x - i_l)},$$
with degree $d' = d - k$, rate $\rho = \frac{d'}{|D|}$ and error-bound δ. Note that due to our convention $d' = 2^n$, so that '2-adicity' constraint of the FRI protocol is satisfied. The verifier has access to $q(x)$ via oracle access to $f(x)$ and their knowledge of all (i_l, z_l) pairs. If the prover passes the FRI protocol, the verifier accepts, rejecting otherwise.

3) LPC Scheme Security: Below we provide an outline of the security of the above scheme.

Completeness: If the prover starts with some $g \in R[X]$ within the δ radius from the exact polynomial f of degree less than $d+k$, then by the completeness of the FRI protocol he would definitely pass the FRI check and the MultiEval method would verify for prover responses with $g(i_l) = z_l$.

Soundness: We claim that the only source of soundness error comes from the FRI protocol, which is inherent to the scheme. We concern ourselves with the situation when $q(x)$ passes the FRI check and the verifier is convinced that $q(x)$ is δ-close to *some polynomial* $h(x)$ with $\deg(h) < d$. This implies that on D:

$$\frac{f(x) - U(x)}{\prod_{j=1}^{k}(x - i_j)} = h(x), \text{ except for } \delta\text{-fraction of points,}$$

$$f(x) = U(x) + h(x) \prod_{j=1}^{k}(x - i_j), \text{ except for } \delta\text{-fraction of pts.}$$

Note that $t(x) = U(x) + h(x) \prod_{j=1}^{k}(x - i_j)$ is a polynomial of degree less than $d + k$. From the second equation we get that this polynomial is δ-close to $f(x)$ or that $\Delta(f, t) < \delta$ where Δ denotes Hamming distance. Moreover, we have that $\forall j \in k$, $t(i_j) = U(i_j) = z_j$ by the definition of $U(i_j)$. This means $t(x)$ satisfies all the requirements for $\mathcal{R}^{\delta}(\mathbb{F}, D, \rho, k)$ stated above.

Remark 1: The prover may efficiently decode every list element (and hence $t(x)$ from above) from representation of $f(x)$ using the Sudan–Guruswami list Decoding Algorithm [33].

Remark 2: Unlike [15], we do not require the scheme to be hiding. We say that a commitment scheme has the *perfectly hiding* property if an opening at any point doesn't give the verifier any additional information about that polynomial apart from its value at the opening. Certainly, if the verifier has collected $l > deg(\phi(x))$ openings to $\phi(x)$ then the polynomial is completely specified and can be constructed via Lagrange interpolation. The reason we do not insist on hiding property for our scheme is that it is not necessary in subsequent sections. We will achieve zero-knowledge on the application level rather that at the level of polynomial commitments.

VI. Polynomial Evaluation Scheme

The list commitment scheme introduced above works fine when dealing with "witness" polynomials, since we are not concerned with the uniqueness property of our scheme, i.e. we are not interested in the exact polynomial g taken from δ-list of f, evaluations of which are opened during MultiEval. However, extra care should be taken outside of this regime, when we work with *setup* polynomials $c(x)$ encoding the constraint system itself. In this case we want to be sure that the openings provided by prover are indeed the evaluations of the polynomial $c(x)$ itself and not of any random polynomial from $L_\delta(c)$.

The simplest approach would be to abolish the use of our commitment scheme for setup polynomials and let the verifier to evaluate setup polynomials on behalf of herself. However, for zk-scheme to be truly succinct, we ideally want the verifier to avoid this task and to simply receive the evaluations of constraint polynomials from the prover (possibly with some short signature of correctness).

With the above in mind, we propose the following workaround. We leverage the fact that for a given setup polynomial $c(x)$ the list $L_\delta(c)$ is theoretically known at the setup phase by both prover and verifier. They can hence find a *distinguisher* point i in which the evaluation of setup polynomial is different from the evaluations of *all* other polynomials in the list. This can be naively achieved by running a list-decoding algorithm once at the beginning to find all $g \in L_\delta$ and then start picking such a point i at random (and checking that $c(i) \neq g(i)\ \forall\ g \in L_\delta$) until we find a suitable candidate. This, however, comes with significant (polynomial in $|D|$) overhead.

The key to our approach is that the procedure of enumerating all such elements and picking a suitable candidate is (1) fully transparent, and (2) executed and verified only once for every circuit. We thus add an *offline phase* that is executed only once at the beginning of the protocol. The task of offline phase is to search for aforementioned distinguisher point i. This allows us to strengthen the proof of knowledge guarantee for the list commitment scheme to imply that all evaluations come from the *specific* polynomial $c(x)$. This permits us to use the list commitment construction with constraint polynomials as well (with minor changes). This *preprocess phase* in completely analogous to the work of *indexer* in terms of [35].

1) Polynomial Evaluation Scheme Semantics:

1) **Setup:** The prover and verifier agree on the following parameters: field $\mathbb{F}$, domain $D \subset \mathbb{F}$, $\delta > 0$ (error-bound), $d \in \mathbb{N}$ (degree of polynomial) and k (the number of points to open).
2) **Preprocess phase:** Prover and verifier agree on a setup polynomial $c(x)$, separation point $i \in \mathbb{F}$ and value $z = c(i)$ with the following properties:

$$\forall g(x) \in L_\delta(c) : g(i) \neq c(i)$$

3) **First message format:** The verifier chooses and sends to the prover a set of k points: $\{i_j\}_{j=1}^{k}$.
4) **Second message format:** The prover responds with values $\{z_j\}_{j=1}^{k}$, which are the purported openings for the $\{i_j\}_{j=1}^{k}$.
5) **Subsequent interaction:** The following interaction is completely analogous to MultiEval interactive oracle protocol defined in the previous section except for the fact that we explicitly add the pair (i, z) to our point-values pairs (i.e. MultiEval is now played w.r.t the sets $\{i, i_1, \ldots, i_k\}$ and $\{z, z_1, \ldots, z_k\}$.)

2) Soundness: We provide soundness analysis for FRI-based instantiation of evaluation scheme and MultiEval protocol.

We begin in a familiar setting: the protocol accepts only if q passes the FRI check and the verifier is convinced that $q(x)$ is δ-close to *some polynomial* $h(x)$ with $\deg(h) < d$. This implies that on D, except for a δ-fraction of points:

$$\frac{c(x) - U(x)}{(x-i)\prod_{j=1}^{k}(x - i_j)} = h(x),$$

$$c(x) = U(x) + h(x)(x-i)\prod_{j=1}^{k}(x - i_j).$$

Note that $t(x) = U(x) + h(x)(x-i)\prod_{j=1}^{k}(x-i_j)$ is a polynomial of degree less than $d + k + 1$. From the second equation we get that this polynomial is δ-close to $c(x)$ or that $\Delta(c,t) < \delta$ where Δ denotes Hamming distance. We have that $\forall j \in k$, $t(i_j) = U(i_j) = z_j$ by the definition of $U(i_j)$. Moreover, the same holds for $t(i) = U(i) = z$. This implies that $t = c$ since $\Delta(c,t) < \delta$ (so $t \in L_\delta(c)$) and $c(i) = t(i)$ where i is our separation point.

Remark: The degree l of constraint polynomial $c(x)$ is usually defined at the beginning by constraint system itself and therefore can't be easily changed. However, our instantiation of polynomial evaluation protocol only works with polynomials of degree $d = 2^n + k + 1$ (the last term comes from the introduction of a new point-value pair (i,z)). This is not a burdensome restriction as we may always take suitable $n' \in \mathbb{N}$ so that $d' = 2^{n'} + k + 1 > l$ and consider our polynomial $c(x)$ as a polynomial in $\mathbb{F}_{<d'}[x]$. Note, that in this case we consider $L_\delta(c)$ to be a subset of $\mathbb{F}_{<d'}[x]$ (and not of $\mathbb{F}_{\leq l}[x]$) This remark should be taken into account during the search for separator point i.

VII. RedShift

A. Constraint system

We start with a description of the constraint system used in PLONK [23] which is then used to design RedShift. We then describe an equivalent encoding of this system in polynomial form.

Constraint system version 1:

The constraint system $\mathcal{L} = (\mathcal{V}, \mathcal{Q})$ with n gates and m wires is defined as follows:

- $\mathcal{V}$ is of the form $\mathcal{V} = (\mathbf{a}, \mathbf{b}, \mathbf{c})$, where $\mathbf{a}, \mathbf{b}, \mathbf{c} \in [m]$
- We call $\mathbf{q_L}, \mathbf{q_R}, \mathbf{q_O}, \mathbf{q_M}, \mathbf{q_C}$ "selector" vectors (*left*, *right*, *output*, *multiplication* and *constant* respectively), and set:

$$\mathcal{Q} = (\mathbf{q_L}, \mathbf{q_R}, \mathbf{q_O}, \mathbf{q_M}, \mathbf{q_C}) \in \mathbb{F}^n.$$

We say $\mathbf{x} \in \mathbb{F}^m$ satisfies $\mathcal{L}$ if for each $i \in [n]$,

$$(\mathbf{q_L})_i \cdot \mathbf{x}_{\mathbf{a}i} + (\mathbf{q_R})_i \cdot \mathbf{x}_{\mathbf{b}i} + (\mathbf{q_O})_i \cdot \mathbf{x}_{\mathbf{c}i} + \\ + (\mathbf{q_M})_i \cdot (\mathbf{x}_{\mathbf{a}i}\mathbf{x}_{\mathbf{b}i}) + (\mathbf{q_C})_i = 0.$$

To define a relation based on $\mathcal{L}$, we extend it to include a positive integer $l \leq m$, and subset $I \subset [m]$ of "public inputs". Without loss of generality, we assume that $I = 1, \ldots, l$.

Define the relation $\mathcal{R}_\mathcal{L}$ as the set of pairs (x, ω) with $x \in \mathbb{F}^l, \omega \in \mathbb{F}^{m-l}$ such that $\mathbf{x} := (x, \omega)$ satisfies $\mathcal{L}$. We say $\mathcal{L}$ is prepared for l public inputs if for $i \in [l]$

$$\mathbf{a_i} = i,\ (\mathbf{q_L})_i = 1,\ (\mathbf{q_M})_i = (\mathbf{q_R})_i = (\mathbf{q_O})_i = (\mathbf{q_C})_i = 0.$$

From here on, we assume the constraint system is in prepared form.

Constraint system version 2:

In order to reformulate this constraint system in polynomial terms we need a bunch of additional definitions. Let $g \in \mathbb{F}^*$ be an element of order $n+1$. Let $H = \{e = g^0, g, g^2, \ldots, g^n\}$ be a cyclic subgroup of $\mathbb{F}^*$ generated by g. Let $H^* = H/\{e\}$. For $i \in [n+1]$ we denote by $L_i(X)$ the element of $\mathbb{F}_{\leq n+1}[X]$ with $L_i(g^i) = 1$ and $L_i(a) = 0$ for $a \in H$ different from g^i, i.e. $L_i(x)$ for $i \in [n+1]$ form a Lagrange basis for H. Define $Z(x) = \prod_{a \in H^*}(X - a)$ be a domain polynomial for H^*, which is zero on all points $a \in H^*$ (and only on them).

Definition 5. *permutation across several polynomials over domain H^*. Suppose we have multiple polynomials $f_1, \ldots, f_k \in \mathbb{F}[X]$ and a permutation $\sigma : [kn] \to [kn]$. For $(h_1, \ldots, h_k) \in (\mathbb{F}[X])_k$, we say that $(h_1, \ldots, h_k) = \sigma(f_1, \ldots, f_k)$ if the following holds. Define the sequences $(f_{(1)}, \ldots, f_{(kn)}), (h_{(1)}, \ldots, h_{(kn)}) \in \mathbb{F}^{kn}$ by*

$$f_{((j-1)\cdot n+i)} := f_j(g^i),\ h_{((j-1)\cdot n+i)} := h_j(g^i),$$

for each $j \in [k], i \in [n]$. Then we should have $h_{(l)} = f_{(\sigma(l))}$ for each $l \in [kn]$.

Definition 6. *Let $\mathcal{T} = T_1, \ldots, T_s$ be a partition of $[kn], k, n \in \mathbb{N}$ into disjoint blocks. We say that $f_1, \ldots, f_k \in \mathbb{F}[X]$ copy-satisfy $\mathcal{T}$ if, when defining $(f_{(1)}, \ldots, f_{(kn)}) \in \mathbb{F}^{kn}$ as above, we have $f_{(l)} = f_{(l')}$ whenever l, l' belong to the same block of $\mathcal{T}$.*

Define a permutation $\sigma(\mathcal{T})$ on $[kn]$ such that for each block T_i of $\mathcal{T}$, $\sigma(\mathcal{T})$ contains a cycle going over all elements of T_i and only over them. There are several possible choices of such a permutation (for example, we can rearrange elements in the cycles corresponding to T_i), $\sigma(\mathcal{T})$ can be taken arbitrary from the set of all allowed permutations. It is simple to check that $(f_1, \ldots, f_k)$ copy-satisfy $\mathcal{T}$ if and only if $(f_1, \ldots, f_k) = \sigma(f_1, \ldots, f_k)$.

The constraint system $\mathcal{L}' = (\mathbf{q_L}, \mathbf{q_R}, \mathbf{q_O}, \mathbf{q_M}, \mathbf{q_C}, \sigma)$ for domain H^* of size n is defined as follows:

1) $\mathbf{q_L}, \mathbf{q_R}, \mathbf{q_O}, \mathbf{q_M}, \mathbf{q_C} \in \mathbb{F}[X]$ - selector polynomials.
2) σ -permutation in $[3n]$ elements.

We define the relation $\mathcal{R}_{\mathcal{L}'}$ as the set $(x, \omega) = (\mathbf{PI}(x), \langle \mathbf{f_L}(x), \mathbf{f_R}(x), \mathbf{f_O}(x) \rangle) \in (\mathbb{F}[X])^4$ with the following properties:

1) $\mathbf{f_L}(x), \mathbf{f_R}(x), \mathbf{f_O}(x)$ copy-satisfy σ.
2) $\forall a \in H^* : \mathbf{q_L}(x) \cdot \mathbf{f_L}(x) + \mathbf{q_R}(x) \cdot \mathbf{f_R}(x) + \mathbf{q_O}(x) \cdot \mathbf{f_O}(x) + \mathbf{q_M}(x) \cdot \mathbf{f_L}(x) \cdot \mathbf{f_R}(x) + (\mathbf{q_C}(x) + \mathbf{PI}(x)) = 0$.

$\mathbf{PI}(x)$ is called *public input* polynomial and encodes public data, $\mathbf{f_L}(x), \mathbf{f_R}(x), \mathbf{f_O}(x)$ are called *left, right and output wires* polynomial respectively and encode prover-only private data.

Conversion between constraint systems:

Here we show a polynomial time transition from the first constraint system to the second. It is easy to check that such a transition can be reversed. Hence, it is enough to construct a proof system for the second relation only.

Suppose $\mathcal{V} = (\mathbf{a}, \mathbf{b}, \mathbf{c})$; think of $\mathcal{V}$ as a vector in $[m]^{3n}$. For $i \in [m]$, let $T_i \subset [3n]$ be the set of indices $j \in [3n]$ such that $\mathcal{V}_j = i$. Now define $T_{\mathcal{L}} := \{T_i\}_{i \in [m]}$ - partition of $[3n]$ into non-intersecting chunks. Define a permutation $\sigma(T_{\mathcal{L}})$ on $[3n]$ in the following way: for each block T_i of $T_{\mathcal{L}}$, $\sigma(T_{\mathcal{L}})$ contains a cycle going over all elements of T_i. For simplicity we write $\sigma = \sigma(T_{\mathcal{L}})$

Overloading notation, set the selector polynomials $\mathbf{q_L}, \mathbf{q_R}, \mathbf{q_O}, \mathbf{q_M}, \mathbf{q_C} \in \mathbb{F}[X]$ defined for each $i \in [n]$ by

$$\mathbf{q_L}(g^i) := (\mathbf{q_L})_i,\ \mathbf{q_R}(g^i) := (\mathbf{q_R})_i, \mathbf{q_O}(g^i) := (\mathbf{q_O})_i,$$

$$\mathbf{q_M}(g^i) := (\mathbf{q_M})_i,\ \mathbf{q_C}(g^i) := (\mathbf{q_C})_i.$$

If (x, ω) is a relation $\mathcal{L}$ prepared for l public inputs, then $(x'\omega')$ is a relation for $\mathcal{L}'$ computed in the following way:

1) $\mathbf{PI}(\mathbf{X}) := \sum_{i \in [l]} -x_i \cdot L_i(X)$
2) $\mathbf{f_L}, \mathbf{f_R}, \mathbf{f_O} \in \mathbb{F}[X]$ are defined by the following condition: $\forall i \in [n]$

$$\mathbf{f_L}(i) = \mathbf{x_{a_i}},\ \mathbf{f_R}(i) = \mathbf{x_{b_i}},\ \mathbf{f_O}(i) = \mathbf{x_{c_i}}.$$

Remark 1: Note that calculation of x' requires only the access to statement x and no access to secret witness ω.

Remark 2: Note that permutation σ was chosen in such a way that ω is a valid witness for $\mathcal{L}|_x$ iff $\mathbf{f_L}, \mathbf{f_R}, \mathbf{f_O}$ constructed as described before from a valid witness for $\mathcal{L}'|_{x'}$.

Remark 3: Note, that "true" degree of polynomials $\mathbf{f_L}, \mathbf{f_R}, \mathbf{f_C}$ is $n-1$ where n is number of gates in $\mathcal{L}$. However in RedShift we will allow them to be of some degree $n' > n$, where the particular choice of n' will be described later.

B. Protocol

Preprocessing: Let $\mathcal{L}' = (\mathbf{q_L}, \mathbf{q_R}, \mathbf{q_O}, \mathbf{q_M}, \mathbf{q_C}, \sigma)$ be constraint system in question.

Take $k_1 = e, k_2, k_3 \in \mathbb{F}^*$ to be representatives of different cosets in $\mathbb{F}^*/H$. Let τ be the bijection between the sets $P_1 = [3n]$ and $P_2 = H^* \cup k_2H^* \cup k_3H^*$ defined by:

$$\tau[n \cdot (j-1) + i] = k_j g^i,\ i \in [n], j \in [3].$$

Recall that σ is a permutation on P_1 hence $\sigma' = \tau \circ \sigma \circ \tau^{-1}$ is a permutation on P_2.

Define $S_{id_1}(X), S_{id_2}(X), S_{id_3}(X), S_{\sigma_1}(X), S_{\sigma_2}(X), S_{\sigma_3}(X) \in \mathbb{F}_{<n}[X]$ - "permutation" polynomials by the following rules:

1) $S_{id_j}(X) = k_j X$ for $j \in [3]$.
2) $S_{\sigma_j}(g^i) = \sigma'(k_j g^i)$ $i \in [n], j \in [3]$.

Remark: Selectors $\mathbf{q_L}, \mathbf{q_R}, \mathbf{q_O}, \mathbf{q_M}, \mathbf{q_C}$, permutation polynomials S_{id_1}, S_{id_2}, S_{id_3}, S_{σ_1}, S_{σ_2}, S_{σ_3} and Langrange-basis polynomials $\{L_i\}_{i \in [n+1]}$ play the role of "constraint" polynomials in terms of previous section.

Setup: Fix FRI parameters and degree d for FRI games (that is, the degree of all quotient polynomials). The prover is given an explicit representation of all constraint polynomials and the verifier is given oracle access to them alongside the "distinguishing" point z (which in general is different for each constraint polynomial). The verifier is given $PI(x)$ - the public inputs polynomial and the honest prover is additionally given $\mathbf{f_L}, \mathbf{f_R}, \mathbf{f_O}$, the witness-wire polynomials.

All commitments mean FRI-commitments in this section.

Protocol:

1) Prover chooses masking polynomials $h_1(x), h_2(x), h_3(x) \in \mathbb{F}_{<k}[x]$ where the choice of k will be described later. Prover defines new witness polynomials $f_1(x) = \mathbf{f_L}(\mathbf{x}) + h_1(x)Z(x), f_2(x) = \mathbf{f_R}(\mathbf{x}) + h_2(x)Z(x), f_3(x) = \mathbf{f_O}(\mathbf{x}) + h_3(x)Z(x)$.
2) Prover sends commitments to polynomials f_1, f_2, f_3 to verifier.
3) Verifier chooses random $\beta, \gamma \in \mathbb{F}$ and sends them to prover.
4) For $j \in [3]$ prover computes $p_j := f_j + \beta \cdot S_{id_j} + \gamma$ and $q_j = f_j + \beta \cdot S_{\sigma_j} + \gamma$. He then defines $p'(X)$ and $q'(X) \in \mathbb{F}_{<3n}[X]$ by

$$p'(X) = \prod_{j \in [3]} p_j(X),\ q'(X) = \prod_{j \in [3]} q_j(X).$$

Prover computes polynomial $P(x), Q(x) \in \mathbb{F}_{<n+1}[X]$ such that $P(g) = Q(g) = 1$ and for $i \in \{2, \ldots, n+1\}$:

$$P(g^i) = \prod_{1 \le j < i} p'(g^j).$$

$$Q(g^i) = \prod_{1 \le j < i} q'(g^j).$$

Prover sends commitments to P and Q.

5) Verifier sends random $a_1, \ldots, a_6 \in \mathbb{F}$ to Prover.
6) Define the following polynomials:
   a) $F_1(x) = L_1(x)(P(x) - 1)$
   b) $F_2(x) = L_1(x)(Q(x) - 1)$
   c) $F_3(x) = P(x)p'(x) - P(xg)$
   d) $F_4(x) = Q(x)q'(x) - Q(xg)$
   e) $F_5(x) = L_n(x)(P(xg) - Q(xg))$
   f) $F_6(x) = \mathbf{q_L}(x) \cdot f_L(x) + \mathbf{q_R}(x) \cdot f_R(x) + \mathbf{q_O}(x) \cdot f_O(x) + \mathbf{q_M}(x) \cdot f_L(x) \cdot f_R(x) + (\mathbf{q_C}(x) + PI(x))$

   Later we show that for honest provers all of $\{F_i(x)\}$ are identically zero on domain H^*. This means that all of $\{F_i(x)\}$ are divisible by $Z(x)$ in the ring $\mathbb{F}[x]$, hence so is their linear combination $F(x) = \sum_{i=1}^{6} a_i F_i(x)$. Prover computes $T(x) = \frac{F(x)}{Z(x)}$ and sends the verifier a commitment to $T(x)$.

   Remark: Due to the restrictions on the degrees it may be necessary to split $T(x)$ into separate polynomials $T_0(x), T_1(x), \ldots, T_3(x)$ and commit to them independently.
7) Verifier uniformly random chooses point $y \in \mathbb{F}/H$ and queries openings for all setup and witness polynomials

at this point. Note that we use the evaluation scheme of section VI to open constraint polynomials and the list version of commit-reveal algorithm of section V to open witness polynomials. By using queried values verifier is able to compute $\{F_i(y)\}_{i \in [6]}$ and $T(y)$.

Remark: We deprecate sampling y inside domain H in order to achieve perfect-zero knowledge instead of statistical.

8) Verifier checks the identity:

$$\sum_{i=1}^{6} a_i F_i(y) = Z(y)T(y) \ \ (*)$$

If this equation holds he accepts the proof and rejects otherwise.

For brevity, we defer a full analysis of the security and zero-knowledge properties of the above scheme to the appendix.

Remark 1: Polynomials $F_i(x)_{i \in [5]}$ are responsible for checking copy-satisfiability of witness polynomials.

Remark 2: Here we explain the intuition behind S_{id_k}, S_{σ_j}. S_{id_k} is only required to map H to disjoint sets P_1, P_2, P_3. S_{σ_j} should then map to the same set $P = P_1 \cup P_2 \cup P_3$ but in a "permuted" fashion. We construct a map τ in order to transfer permutation σ from domain $[n]$ to P. The simplest way to define S_{id_k} is to map $[n]$ to $[1..n], [n+1..2n], [2n+1, 3n]$ respectively, in this case there is no need to apply the map τ as then there is no need of domain translation ($P = [n]$). The problem is that all of the S_{id_k} polynomials will be of degree n in general. We construct S_{id_k} in such a way to be of minimal possible degree - 1, so it is easy to verifier to calculate evaluations of those polynomials by himself and we get rid of "polynomial evaluation protocols" in this case.

Remark 4: Although by described construction $\mathbf{f_L}, \mathbf{f_R}, \mathbf{f_O}$ can be taken to be in $\mathbb{F}_{<n}[X]$, the concrete degrees of them are not important: the only required property is the relation between these values on domain H^*. This freedom in degree choice helps achieve zero-knowledge.

VIII. Optimizations and variants

In this section we are going to describe various additional techniques to achieve better concrete efficiency of the RedShift protocol.

A. Batched FRI

Recall that all polynomial commitments and evaluations in our protocol are reduced via FRI to the following check: whether particular functions $f_1, \dots, f_k$ represented as oracles are close to the space of degree d polynomials. The intuitive approach is to replace all those separate and independent FRI queries by exactly one instance of FRI w.r.t a linear combination of functions f_i, where the coefficients of linear dependence are provided by the verifier. This can in fact be done, with rigorous justification based on the following lemma found in [27].

Definition 7. *$J_\epsilon^{[k]}(\lambda) = J_\epsilon(J_\epsilon(\cdots(J_\epsilon(\lambda))))$, where there are k iterations of the function J_ϵ.*

Definition 8. *The relative hamming distance of set $S \subseteq \mathbb{F}^n$ is $\Delta(S) = min\{\Delta(w, w_0) | w, w_0 \in S, w \neq w_0\}$.*

Theorem 5. *Let $V \subseteq \mathbb{F}^n$ be a linear space over a finite field $\mathbb{F}$ with $\Delta(V) = \lambda$. Let $u^* \in \mathbb{F}^n$ and let $\epsilon > 0$ satisfy $\delta < J_\epsilon^{[l+1]}(\lambda)$. For $u_1, u_2, \dots, u_l \in \mathbb{F}^n$ let $A = \{\alpha \in \mathbb{F}^* / |\Delta(u^* + \alpha u_1 + \alpha^2 u_2 + \cdots + \alpha^l u_l, V) < \delta\}$. If $|A| > l \cdot \left(\frac{2}{\epsilon}\right)^{l+2}$, then there exist $v^*, v_1, v_2, \dots, v_l \in V$ such that*

$$|\{i \in [n] \mid (u_i^* = v_i^*) \wedge ((u_1)_i = (v_1)_i) \wedge \cdots \wedge ((u^l)_i = (v^l)_i)\}$$

is $\geq (1 - \delta - \epsilon)n$. In particular, $\Delta(u^*, v^*) \leq \delta + \epsilon$ and $\forall i \in [l] : \Delta(u_l, v_l) \leq \delta + \epsilon$.

Specifying this theorem for $V = \mathsf{RS}[\mathbb{F}, D, \rho]$, for which $\lambda = \Delta(V) = 1 - \rho$. We use the contrapositive to get the following corollary:

Corollary 1. *Let $V = \mathsf{RS}[\mathbb{F}, D, \rho]$ be the family of RS-codes. Let $\epsilon \in (0,1), \delta > 0$ are chosen is such a way that $\delta < J_\epsilon^{[l]}(1 - \rho)$. Let $l \geq 2 \in \mathbb{N}$ and $u_1, u_2, \dots, u_l \in \mathbb{F}^n$, such that there exists $i \in [l]$ for which $\Delta(u_i, V) > \delta + \epsilon$. Then $|A| \leq (l-1)\left(\frac{2}{\epsilon}\right)^{l+1}$.*

We have the following protocol for a batched FRI, the correctness of which is a trivial consequence of the previous corollary.

Batched FRI protocol:

1) Prover publishes oracles to $f_1, \dots, f_k$.
2) Verifier selects random $\alpha \in \mathbb{F}^*$ and send it to prover.
3) Prover and Verifier are engaged into FRI-protocol w.r.t $f = \sum_{i=1}^{k} \alpha^{i-1} f_i$. Verifier's output is *acc* if prover has passed FRI check and *rej* otherwise.

B. Remarks on the Setup Phase

Recall that as part of the setup for RedShift, for every constraint polynomial c we are required to find a specific point $i \in \mathbb{F}$ which separates c from all polynomials taken the from corresponding δ-list $L_\delta(c)$. As we have noted in previous sections, decoding the whole list can be conducted in a polynomial number of steps (through the Sudan decoding-algorithm) and such a point i can be quickly found by brute-force. Despite the fact that all these preliminary actions should be done only once per circuit and the resultant i can be subsequently used in all created proofs, such a setup can be a significant amount of work. A possible workaround for this problem is to forgo decoding algorithms altogether and randomly sample point i, e.g. as a hash of the circuit using techniques similar to Fiat-Shamir heuristics. Due to the Schwartz-Zippel lemma, there is a high chance that i will indeed separate c from the corresponding δ-list. If i is taken in a way that polynomials are indistinguishable (so some other polynomial from the list $L_\delta(c)$ has the same value at the point i and polynomial c) then the malicious prover will be able to conduct proof forgery. As a result we have a trade-off between speed, efficiency and security of the whole protocol. We calculate the extra soundness penalty due to this below.

If at the setup step the choice of $i \in \mathbb{F}$ was random, the probability that any two degree d polynomials $g_1, g_2 \in L_\delta(c)$ satisfy $g_1(i) = g_2(i)$ is small. To see this, by the Schwartz-Zippel lemma we have that $\forall g \in L_\delta(c)$:

$$\Pr_i[g(i) - c(i) = 0] \leq \frac{\deg(g(X) - c(X))}{|\mathbb{F}|} \leq \frac{d}{|\mathbb{F}|}$$

Enumerating over all $g_j \in L_\delta(c) \backslash \{c\}$ where $j \in |L_\delta(c)|$, from a union bound we get that:

$$\Pr_i\left[\bigcup_{j \in |L_\delta|-1} g_j(i) - c(i) = 0\right] \leq \frac{d}{|\mathbb{F}|} \cdot (|L_\delta| - 1)$$

1) Improving soundness with multiple samples: One issue with the above lies in the size of $|L_\delta|$. In cases where this is too large, one random sample might not be enough to achieve the desired soundness bound.

We show that the straightforward approach of sampling *multiple* random points yields substantial soundness improvements. In this case the verifier samples k random points $\{i_j\}_{j=1}^k$ computed the corresponding values $\{c(i_j)\}_{j=1}^k$ at the setup phase. The prover then performs an opening at the point $i \in \mathbb{F}$ by performing an IOPP for membership of the following function in $\mathsf{RS}[\mathbb{F}_q, D, d/|D|]$:

$$h(X) = \frac{c(X) - U(X)}{(X - i)\prod_{j=1}^k (X - i_j)}$$

where $U(X)$ interpolates between the target opening point (i, z) and all other points $(i_j, c(i_j))$.

Given that the $i_j \in \mathbb{F}$ are randomly sampled, then the probability that some $g \in L_\delta(f)$ agrees with c on *all* k points is actually substantially smaller:

$$\Pr_{i_1,\ldots,i_k}\left[\bigcap_{j \in [k]} g(i_j) - c(i_j) = 0\right] =$$
$$= \prod_{j=1}^k \Pr_i[g(i) - c(i) = 0] \leq \left(\frac{d}{|\mathbb{F}|}\right)^k.$$

From the union bound, we get the total soundness error to be equal to $|L_\delta(c)| \cdot (d/|\mathbb{F}|)^k$.

Limitations on k: It is immediate that $k \leq d$. Indeed, the case where $k = d$ fully specifies the polynomial since f is of degree d. Sampling and evaluation procedure is $O(kd)$ in terms of computation complexity and has to be done only once.

C. Binary fields

Recall that Kate-based PLONK is restricted to prime fields only. The reason for this is that the Kate commitment requires embedding a field $\mathbb{F}$ into a group of points on some pairing-friendly elliptic curve. Such an embedding is known for prime fields $\mathbb{F}$ only. In [27] there is a version of the FRI protocol for binary fields which is similar to the one used here but which exploits additive and vector space structure of the underlying field instead of the multiplicative one. All other parts of PLONK are completely field agnostic and only permutation argument may require modifications. This means that RedShift can be instantiated for binary fields as well as prime fields and all constructions and proofs follow through: simply replace the multiplicative domain $|D|$ by an affine subspace. The binary variant of PLONK is especially effective for computations that require a lot of XOR's and bit manipulations, which are naturally encoded in a binary version of the system.

D. Recursion

One can express a verification subroutine of RedShift as another circuit, where the dominating subroutine will be verification of Merkle paths (if the IOP is instantiated with Merkle trees), or inclusion proofs in some other cryptographic accumulator (e.g. RSA-based). All the remaining arithmetic operations are performed over the same field that the original circuit (for which verifier is expressed) is defined, so there is no requirement for cycles over pairing friendly elliptic curves as in previous work. We should also note that one can use a hybrid approach to perform the last step of recursion using a pairing-based PLONK, e.g. the BLS12-381 curve has a main subgroup of order $|G|$ such that $2^{32} \mid (|G| - 1)$. This allows us to instantiate RedShift.

E. Application to other proof systems

One can apply the list polynomial commitment scheme and evaluation scheme to other proof systems such as SONIC [14] and Marlin [35] that were also originally instantiated using univariate polynomial commitments. We leave such transformation for an interested reader and will only suggest that one will have to pay more attention to proximity testing parameters e.g. testing inclusion in $\mathsf{RS}[\mathbb{F}_q, D, (d-1)/|D|]$, where $d = 2^k$ in the case of Marlin. [29] contains a description of such a subroutine.

We should also note that the DEEP-ALI protocol from [28] uses a construction that is equivalent to the list commitment scheme in their application to the STARK [18] proof system. In this case, all the setup polynomials, constraints and checked relations are known to the verifier and are checked naively during the verification procedure. We expect that our polynomial evaluation scheme can also be adapted for this case and allow one to express more complex STARK circuits.

F. Strict commitment scheme

An interested reader may have noticed that all the logic about list commitment schemes can be combined with a requirement that the FRI δ parameter is in the unique decoding radius. This provides a standard polynomial commitment scheme. Introducing the limitation of unique decoding also lifts the requirement for a separate evaluation scheme. Such a primitive was already described in [36] and as mentioned in the introduction it comes at the cost of larger proof sizes.

IX. System instantiation

Now, we have constructed and proved RedShift from several components based on various independent parameters: ρ, ϵ, δ, n. In this section we want to come to common base and give a concrete example how one can instantiate all the parameters of the system.

First we focus on the contribution $A' = L^{t_2}\frac{4n}{|\mathbb{F}/D|}$, for which the full soundness analysis can be found in Appendix B. Following theorem 1, we choose $\epsilon = |\mathbb{F}|^{-1/20}$. This provides a list size of $O(\mathbb{F}^{1/20})$, and for any reasonable problem size n we have $t_2 = 8$ witness polynomials (enumerated later in section X) and $A' \sim 1/\sqrt{\mathbb{F}}$. For a typical field size of ~ 256 bits, the error contribution due to A' is on the order of 2^{-128}.

Now we focus on how such choice of $\epsilon \neq 0$ will affect the formula of FRI soundness from section IV. We choose $\rho = 1/16$ and are interested in the part of the formula involving

$$p = \Big(1 - min\Big\{\delta_0, J_\epsilon(J_\epsilon(1-\rho))\Big\} + \epsilon \log|D|\Big).$$

A smaller value of p allows one to have fewer queries to achieve the same FRI soundness error. In the case of $\epsilon = 0$, we have $p_0 = 1/2$ for our choice of ρ. For $\epsilon = \mathbb{F}^{-1/20}$ and domain size $|D| = 2^{32}$ this is equivalent to $n+1 = 2^{28}$ and we have that $p \sim 0.504$.

If we instead use $\rho = 1/32$, then for the same choice of the other parameters $p_0 \sim 0.421$ and $p \sim 0.425$. In all the cases $\delta = J_\epsilon(J_\epsilon(1-\rho))$ is below the Johnson bound $1-\sqrt{\rho}$.

Here we should also give numeric estimates for benefit of moving away from unique decoding radius and using list commitment scheme instead of testing for unique polynomial. For a rate of $\rho = 1/32$ putting $\epsilon = 0$ for ease of illustration and aiming for 80 bits of security we have the following comparison

1) $\delta = \delta_0 = \frac{1-\rho}{2}$ results in $p_0 = 1 - \delta_0 = \frac{33}{64}$ and required number of queries is ~ 84
2) Using FRI [34] and list commitment $\delta = J_\epsilon(J_\epsilon(1-\rho))$ results in $p = \sqrt[4]{\rho}$ and required number of queries is ~ 64
3) Using DEEP-FRI [28] and list commitment $\delta = J_\epsilon^{3/2}(1-\rho)$ results in $p = \sqrt[3]{\rho}$ and required number of queries is ~ 48

Same estimate for $\rho = 1/64$ would result in values of 82, 52 and 40 queries respectively.

X. Benchmarks

We instantiate RedShift with $q = r \cdot 2^{192} + 1$, $r = 576460752303423505$ which is a Proth prime, and use $\rho = 1/16$. Oracles were instantiated as Merkle trees using the Blake2s hashing algorithm. The setup phase was performed using the approach from Section VIII, where a random point was sampled using Fiat-Shamir heuristics by placing all individual root hashes of oracles to the setup polynomials into the transcript. For comparison, we have also implemented the PLONK prover using Kate commitments and used two curves: BN254 and BLS12-381, that are expected to provide ~ 80 bits and ~ 120 bits of security levels respectively.

Circuit sizes were chosen so as to set the $n+1$ value for PLONK and RedShift to $\left[2^{18}, 2^{19}, ..., 2^{23}\right]$. In the case of RedShift, the degree bound for FRI checks is $d = n+1$. We used a cloud "memory optimized" instance from *DigitalOcean* with 16 (virtual) cores. Results are presented in Fig. 1 with execution time measured in *seconds*.

All the implementations use a certain degree of precomputation: we precompute low degree extensions of setup polynomials (for both RedShift and PLONK) and Merkle trees of the setup polynomial oracles (RedShift only) before proof generation. We also separately provide expected proof sizes for RedShift for different sets of parameters.

We include calculations for approximate proof sizes at different security levels, which are displayed in Table I. A detailed description of how we calculate proof size and applicable optimizations is presented in Appendix D. Results from Table I also hint that in case of recursive constructions one can use different strategies, e.g. use higher rate (ρ) for "inner" level and lower one for levels of recursion that perform a verification of the "inner" proofs.

We note that there exists a specific interplay between the system parameters, running time and proof sizes. As soon as the field $\mathbb{F}$ and parameter ϵ are set, this fixes the list size L and soundness error $A_{SZ} = L^{t_2}\frac{4n}{|\mathbb{F}/D|} + \frac{2}{|\mathbb{F}|}$ of RedShift (as analyzed in Appendix B-A) and the contribution from the randomized setup in VIII-B, if one is applied. Another component of the soundness error is due to the FRI error A_{FRI}, which depends on the number of queries. This *does not* change the proof generation time and only affects proof size. As shown in Section IX, we target a contribution $A_{SZ} \sim 1/\sqrt{|\mathbb{F}|} \sim 2^{-120}$ and then pick the number of queries so that they achieve a specific FRI soundness level. The final soundness error would then be $A_{SZ} + A_{FRI}$, but in some cases one contribution clearly dominates in this sum. This is reflected in Table I where we target security depends from A_{FRI} contribution. We note that the running times in Figure 1 are based on a prover running without all the optimizations implemented, however we believe that incorporation of the optimization techniques will not affect running time but rather provide marginal improvements.

XI. Further work

An immediate follow-up should be the investigation of how the FRI improvement described in [28] affects proving time and final proof sizes. Such a question requires an efficient implementation of the DEEP-FRI protocol. From our previous calculations, we anticipate that the improvements due to the list commitment scheme would be more pronounced for [28] than for [27]. Another important practical aspect is implementation of the verification circuit and explicit testing of recursive

TABLE I: Expected proof sizes for various setup parameters.

Target security level, bits	Rate	Problem size	Proof size, unoptimized	Proof size, optimized
80	1/16	2^{20}	1772 KB	623 KB
80	1/16	2^{24}	2207 KB	761 KB
80	1/16	2^{28}	2682 KB	933 KB
80	1/32	2^{20}	1877 KB	513 KB
80	1/64	2^{20}	1315 KB	465 KB
80	1/128	2^{20}	1179 KB	412 KB
80	1/256	2^{20}	1104 KB	381 KB
120	1/16	2^{20}	2657 KB	935 KB
120	1/16	2^{24}	3309 KB	1141 KB
120	1/16	2^{28}	4022 KB	1400 KB

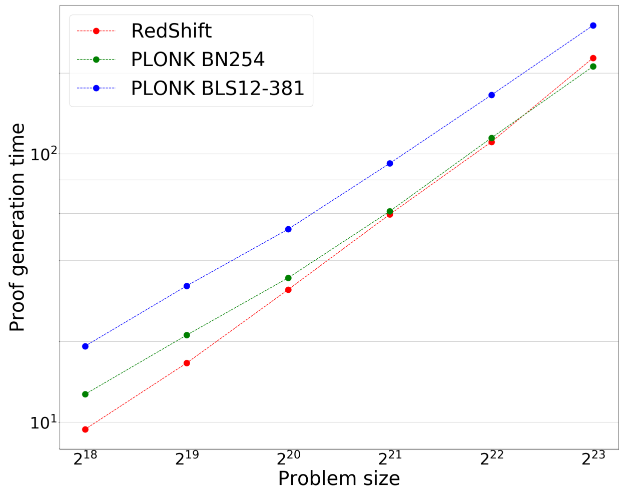

Fig. 1: Benchmark data for settings as described in Section X. Execution time measured in *seconds*.

construction and the resulting circuit size. This would also require investigation of alternative hash functions as described in [37].

As described in Section X, the PLONK proof system initially has 17 different polynomials. The latest updates allow one to reduce this number by one or two, which would also lead to smaller proof sizes and faster proof generation. It is also important to see if an application of RedShift to other existing proof systems would result in even smaller proof sizes for similar circuit sizes or faster proof generation. Another important comparison would be against current state-of-the-art proof systems such as Groth16 [7] with respect to proof generation time. This requires an efficient compiler to transform the R1CS based problem representation to the PLONK arithmetization.

Perhaps a more important question is whether one would need to use the described polynomial evaluation scheme for "constraint" polynomials or if there exist other approaches for how to guarantee the opening of a particular polynomial without the unique decoding radius requirement for FRI.

XII. Acknowledgements

Authors would like to express gratitude to Ethereum Foundation for support of Matter Labs, Eli Ben-Sasson for discussion of FRI, Ariel Gabizon and Zachary J. Williamson for their construction of PLONK and helpful discussion on completeness problem, Sean Bowe and Mary Maller for feedback on initial ideas, Alan Szepieniec for review of the early version of this paper and Dev Ojha for spotting an error in FRI soundness formula.

References

[1] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM Journal on computing*, vol. 18, no. 1, pp. 186–208, 1989.
[2] D. Boneh, J. Bonneau, B. Bünz, and B. Fisch, "Verifiable delay functions," in *Annual International Cryptology Conference*. Springer, 2018, pp. 757–788.
[3] P. Chaidos, V. Cortier, G. Fuchsbauer, and D. Galindo, "Beleniosrf: A non-interactive receipt-free electronic voting scheme," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1614–1625.
[4] S. Setty, S. Angel, T. Gupta, and J. Lee, "Proving the correct execution of concurrent services in zero-knowledge," in *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, 2018, pp. 339–356.
[5] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.
[6] I. Miers, C. Garman, M. Green, and A. D. Rubin, "Zerocoin: Anonymous distributed e-cash from bitcoin," in *2013 IEEE Symposium on Security and Privacy*. IEEE, 2013, pp. 397–411.
[7] J. Groth, "On the size of pairing-based non-interactive arguments," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2016, pp. 305–326.
[8] S. Micali, "Computationally sound proofs," *SIAM Journal on Computing*, vol. 30, no. 4, pp. 1253–1298, 2000.
[9] C. Gentry and D. Wichs, "Separating succinct non-interactive arguments from all falsifiable assumptions," in *Proceedings of the forty-third annual ACM symposium on Theory of computing*. ACM, 2011, pp. 99–108.
[10] R. Gennaro, C. Gentry, B. Parno, and M. Raykova, "Quadratic span programs and succinct nizks without pcps," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2013, pp. 626–645.
[11] S. Setty, B. Braun, V. Vu, A. J. Blumberg, B. Parno, and M. Walfish, "Resolving the conflict between generality and plausibility in verified computation," in *Proceedings of the 8th ACM European Conference on Computer Systems*. ACM, 2013, pp. 71–84.
[12] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, "Succinct non-interactive zero knowledge for a von neumann architecture," in *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, 2014, pp. 781–796.
[13] S. Bowe, A. Gabizon, and I. Miers, "Scalable multi-party computation for zk-snark parameters in the random beacon model." *IACR Cryptology ePrint Archive*, vol. 2017, p. 1050, 2017.
[14] M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn, "Sonic: Zero-knowledge snarks from linear-size universal and updateable structured reference strings." *IACR Cryptology ePrint Archive*, vol. 2019, p. 99, 2019.
[15] A. Kate, G. M. Zaverucha, and I. Goldberg, "Constant-size commitments to polynomials and their applications," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2010, pp. 177–194.
[16] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum, "Delegating computation: interactive proofs for muggles," *Journal of the ACM (JACM)*, vol. 62, no. 4, p. 27, 2015.

[17] R. S. Wahby, I. Tzialla, A. Shelat, J. Thaler, and M. Walfish, "Doubly-efficient zksnarks without trusted setup," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 926–943.

[18] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Scalable zero knowledge with no trusted setup," in *Annual International Cryptology Conference*. Springer, 2019, pp. 701–732.

[19] P. Evgenya, "Algebraic ram," Master's thesis, Technion, 4 2017.

[20] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 315–334.

[21] T. Xie, J. Zhang, Y. Zhang, C. Papamanthou, and D. Song, "Libra: Succinct zero-knowledge proofs with optimal prover computation." *IACR Cryptology ePrint Archive*, vol. 2019, p. 317, 2019.

[22] S. Setty, "Spartan: Efficient and general-purpose zksnarks without trusted setup."

[23] A. Gabizon, Z. J. Williamson, and O. Ciobotaru, "Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge," Cryptology ePrint Archive, Report 2019/953, Tech. Rep., 2019.

[24] B. Bünz, B. Fisch, and A. Szepieniec, "Transparent snarks from dark compilers," Cryptology ePrint Archive, Report 2019/1229, 2019, https://eprint.iacr.org/2019/1229.

[25] E. Ben-Sasson, A. Chiesa, and N. Spooner, "Interactive oracle proofs," in *Theory of Cryptography Conference*. Springer, 2016, pp. 31–60.

[26] A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. Ward, "Marlin: Preprocessing zksnarks with universal and updatable srs," Cryptology ePrint Archive, Report 2019/1047, 2019, https://eprint.iacr.org/2019/1047.

[27] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Fast reed-solomon interactive oracle proofs of proximity," in *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[28] E. Ben-Sasson, L. Goldberg, S. Kopparty, and S. Saraf, "Deep-fri: Sampling outside the box improves soundness," *arXiv preprint arXiv:1903.12243*, 2019.

[29] E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward, "Aurora: Transparent succinct arguments for r1cs," Cryptology ePrint Archive, Report 2018/828, 2018, https://eprint.iacr.org/2018/828.

[30] W. C. Huffman and V. Pless, *Fundamentals of error-correcting codes*. Cambridge: Cambridge Univ. Press, 2003. [Online]. Available: https://cds.cern.ch/record/1139892

[31] E. Berlekamp, "Nonbinary BCH decoding (abstr.)," *IEEE Transactions on Information Theory*, vol. 14, no. 2, pp. 242–242, Mar. 1968. [Online]. Available: https://doi.org/10.1109/tit.1968.1054109

[32] J. Massey, "Shift-register synthesis and BCH decoding," *IEEE Transactions on Information Theory*, vol. 15, no. 1, pp. 122–127, Jan. 1969. [Online]. Available: https://doi.org/10.1109/tit.1969.1054260

[33] V. Guruswami and M. Sudan, "Improved decoding of reed-solomon and algebraic-geometry codes," *IEEE Transactions on Information Theory*, vol. 45, no. 6, pp. 1757–1767, 1999. [Online]. Available: https://doi.org/10.1109/18.782097

[34] E. Ben-Sasson, S. Kopparty, and S. Saraf, "Worst-case to average case reductions for the distance to a code," in *Proceedings of the 33rd Computational Complexity Conference*, ser. CCC '18. Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, pp. 24:1–24:23. [Online]. Available: https://doi.org/10.4230/LIPIcs.CCC.2018.24

[35] A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. Ward, "Marlin: Preprocessing zksnarks with universal and updatable srs," Cryptology ePrint Archive, Report 2019/1047, 2019, https://eprint.iacr.org/2019/1047.

[36] A. Vlasov and K. Panarin, "Transparent polynomial commitment scheme with polylogarithmic communication complexity," Cryptology ePrint Archive, Report 2019/1020, 2019, https://eprint.iacr.org/2019/1020.

[37] A. Chiesa, D. Ojha, and N. Spooner, "Fractal: Post-quantum and transparent recursive proofs from holography," Cryptology ePrint Archive, Report 2019/1076, 2019, https://eprint.iacr.org/2019/1076.

[38] "Starkdex: Bringing starks to ethereum," https://blog.0xproject.com/starkdex-bringing-starks-to-ethereum-6a03fffc0eb7, accessed: 2090-12-05.

APPENDIX A
FRI OVERVIEW

Here we provide an overview of the FRI protocol, as seen in [27].

Definition 9. *For a function $f : S \to \mathbb{F}$, let $\mathsf{interpolant}^f$ be the unique degree $< |S|$ polynomial that satisfies $\mathsf{interpolant}^f(s) = f(s)$ for all $s \in S$. This polynomial can be constructed by Lagrange interpolation.*

Setup phase. In the setup phase, the prover and verifier agree on the following parameters

- A prime field $\mathbb{F}$.
- A positive integer $R \in \mathbb{Z}_{>0}$ and the rate $\rho = 2^{-R}$.
- A multiplicative domain $D = D^{(0)} = \{\omega, \omega^2, \dots, \omega^n\}$ generated by an element $\omega = \omega_0 \in \mathbb{F}^*$ of order $n = 2^k$ for some $k \in \mathbb{N}$. For chosen $\rho = 2^{-R}$ and $n = 2^k$ the FRI protocol will check whether f is of degree $< \rho n = 2^{k-R}$.
- The prover and verifier agree on a number of rounds $r < k - R \in \mathbb{N}$ and a sequence of sub-domains $D^{(0)}, D^{(1)}, D^{(2)}, \dots, D^{(r)}$, constructed inductively as follows. Suppose $D^{(i)}$ was already defined and generated (as a cyclic group) by ω_i. Let $q(x) : \mathbb{F} \to \mathbb{F}$ be the map defined by the rule: $q(x) = x^2$. Then define $D^{(i+1)} = q(D^{(i)})$. Note that $D^{(i+1)}$ is cyclic subgroup of $\mathbb{F}^*$ generated by $\omega_{i+1} = \omega_i^2$. Note that always $|D^{(i)}| = \frac{|D^{(0)}|}{2^i}$ and for all $i \in 0, 1, \dots, r-1$ $D^{(i)}$ can be split into cosets $\cup_j s_{ij} H^{(i)}$ where $H^{(i)}$ is the kernel of the homomorphism $q(x)|_{D^{(i)}} : D^{(i)} \to D^{(i+1)}$. Note that all cosets have equal size $\frac{|D^{(i)}|}{|D^{(i+1)}|} = 2$ and the number of cosets $j = \frac{|D^{(i)}|}{2^{i+1}}$.

When we say that prover commits to function f on domain D this means prover sends an oracle containing $f|_D$ i.e. all evaluations of function f on domain D.

Commit phase. In the commitment phase, the prover inductively constructs and commits to a sequence of functions $f^{(0)}, \dots, f^{(r-1)}$ and a sequence of coefficients $a_0, \dots, a_d$ with which the verifier will construct the final function $f^{(r)}$.

- Input: a purported low degree polynomial $f^{(0)} := f \in \mathsf{RS}[\mathbb{F}, D^{(0)}, \rho]$. The prover commits to $f^{(0)}$ on $D^{(0)}$.
- For $0 \le i < r$, given that $f^{(i)}$ was already defined (and committed to), the prover constructs $f^{(i+1)} : D^{(i+1)} \to \mathbb{F}$ in the following way:
  - The verifier sends a random $x^{(i)} \in \mathbb{F}$.
  - For $y \in D^{(i+1)}$, let $S_y = \{x \in D^{(i)} : q(x) = y\}$ be the coset of $D^{(i)}$ mapped to y.
  - Using interpolation, the prover construct the polynomial
  $$p_y^{(i)}(x) := \mathsf{interpolant}^{f^{(i)}|_{S_y}(x)},$$
  and defines
  $$f^{(i+1)}(y) := p_y^{(i)}(x^{(i)}).$$
- If $i < r - 1$, the prover commits to the values of $f^{(i+1)}$ on $D^{(i+1)}$. If $i = r-1$ then $f^{(r)}$ is a purported polynomial

of degree $< \rho|D^{(r)}|$, in which case the prover commits to its coefficients $a_0, \ldots, a_d$.

Query phase. In the query phase, the verifier (probabilistically) validates the proof sent by the prover.

- ◦ Input: a sequence of oracles $f^{(0)}, \ldots, f^{(r-1)}$, and coefficients $a_0, \ldots, a_d$, with which the verifier constructs $f^{(r)}$, by

$$f^{(r)}(x) := \sum_{k=0}^{d} a_k x^k \in \mathsf{RS}[\mathbb{F}, D^{(r)}, \rho].$$

- ◦ Verifier generates a random $s^{(0)} \in D^{(0)}$ and for all $0 \leq i < r$ lets
  1) $s^{(i+1)} := q(s^{(i)})$
  2) $S^{(i)}$ be the coset of $H^{(i)}$ in $D^{(i)}$ containing $s^{(i)}$.
- ◦ For $0 \leq i < r-1$ the verifier checks that given $f^{(i)}$, the function $f^{(i+1)}$ was constructed according to the protocol:
  - She queries $f^{(i)}$ on all of $S^{(i)}$, and
  - computes $p^{(i)} = \mathsf{interpolant}^{f^{(i)}|_{S^{(i)}}}$, and
  - performs "round consistency" check:

$$f^{(i+1)}(s^{(i+1)}) = p^{(i)}(x^{(i)}).$$

Note that in the last check, the function considered is $f^{(r)}$ which is in $\mathsf{RS}[\mathbb{F}, D^{(r)}, \rho]$ by construction. If all tests pass, the verifier accepts the proof. Otherwise, she rejects.

Remark: Instead of taking a family of nested sub-domains to be multiplicative subgroups it is also possible to take the cosets of them. To be more precise, consider any shift $g \in \mathbb{F}^*/D$. There is a modification to FRI protocol operating over the domains $D^{(0)'} = gD^{(0)}, D^{(1)'} = gD^{(1)}, \ldots, D^{(r)'} = gD^{(r)}$ The function mapping $D^{(i)'}$ to $D^{(i+1)'}$ is $q'(x) = q^{-1}x^2$. The modified version of FRI has the same security guarantees as the original one. The possibility to operate in cosets will be later exploited to achieve zero-knowledge property of transparent PLONK.

Appendix B
RedShift Security Analysis

Here we provide the complete security analysis and zero knowledge proof for RedShift.

A. Security analysis

Completeness: Assume prover posses a valid witness consisting of polynomials $\mathbf{f_L}, \mathbf{f_R}, \mathbf{f_O}$. which copy-satisfy $\mathcal{T}$. Note that addition of masking polynomials doesn't change the values of $\mathbf{f_L}, \mathbf{f_R}, \mathbf{f_O}$ on H, and only the values are checked in the protocol. It is straightforward to check that $F_6(x)$ will be identically zero on H^* by the definition of witness polynomials, $F_1(x), F_2(x), F_3(x), F_4(x)$ will be zero on H^* by construction of $P(x)$ and $Q(x)$. To prove completeness of the protocol it is then enough to check $F_5(x)$ is identically zero on H^*. Using the properties of Lagrange-basis it is equivalent for $P(g^{n+1}) = Q(g^{n+1})$. Using the definition of $P(x)$ and $Q(x)$ the last equation is equivalent to:

$$\prod_{i=1}^{n} \prod_{j=1}^{3} \Big(f_j(g^i) + \beta \cdot k_j g^i + \gamma \Big) =$$

$$= \prod_{i=1}^{n} \prod_{j=1}^{3} \Big(f_j(g^i) + \beta \cdot \sigma'(k_j g^i) + \gamma \Big).$$

Using the definition of τ and $\sigma' = \tau \circ \sigma \circ \tau^{-1}$ we can rewrite this as following:

$$\prod_{i=1}^{n} \prod_{j=1}^{3} \Big(f_{(j-1)n+i} + \beta \cdot \tau\big((j-1)n + i\big) + \gamma \Big)$$

$$= \prod_{i=1}^{n} \prod_{j=1}^{3} \Big(f_{(j-1)n+i} + \beta \cdot \tau \circ \sigma\big((j-1)n + i\big) + \gamma \Big)$$

Now we use the fact $f_1 = \mathbf{f_L}, f_2 = \mathbf{f_R}$ and $f_3 = \mathbf{f_0}$ copy-satisfy $\mathcal{T}$ which means:

$$f_{(j-1)n+i} = f_{\sigma((j-1)n+i)}$$

After renumerating the products on both sides would be completely equal which proves completeness.

B. Soundness and argument of knowledge.

It is enough to prove argument of knowledge property as soundness is an easy corollary of this.

In order to conduct the proof we need three auxiliary statements (the first two are proved in PLONK paper, the last is classic).

Lemma 1. *Let $k \in \mathbb{N}$. Fix $F_1, \ldots, F_k \in \mathbb{F}[X]$. Fix $Z \in \mathbb{F}[X]$. Suppose that for some $i \in [k]$, $Z \nmid F_i$. Then except with probability $\frac{1}{|\mathbb{F}|}$ over uniformly random $a_1, \ldots, a_k \in \mathbb{F}$, $Z \nmid F$, where $F := \sum_{i=1}^{k} a_i F_i$.*

Lemma 2. *Let $n \in \mathbb{N}$. Fix a permutation σ of $[n]$, and $a_1, \ldots, a_n, b_1, \ldots, b_n \in \mathbb{F}$. Suppose that for some $i \in [n]$ $b_i \neq a_{\sigma(i)}$. Then except with probability $\frac{n}{|\mathbb{F}|}$ over random $\beta, \gamma \in \mathbb{F}$:*

$$\prod_{i=1}^{n} (a_i + \beta i + \gamma) = \prod_{i=1}^{n} (b_i + \beta\sigma(i) + \gamma).$$

Lemma 3 (Schwartz-Zippel lemma). *Let $P \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a non-zero polynomial of total degree $d \geq 0$ over a field $\mathbb{F}$. Let S be a finite subset of $\mathbb{F}$ and let $r_1, r_2, \ldots, r_n$ be selected at random independently and uniformly from S. Then*

$$\Pr[P(r_1, r_2, \ldots, r_n) = 0] \leq \frac{d}{|S|}.$$

We will apply last lemma to the special case of univariate polynomial $P(x)$ and the set S being all of $\mathbb{F}$.

We will show that if prover is able to convince verifier in correctness of the statement x then there actually exist an extractor having full access to prover's state and with high probability capable of outputting valid witness $\mathbf{f_L}, \mathbf{f_R}, \mathbf{f_O} \in \mathcal{L}'|_x$ for given statement $x = \mathbf{PI}(x)$.

First, let us count the overall number of FRI instances conducted during the protocol. Let t_1 denote the number of FRI used for evaluation of setup polynomials (exploiting the technique of section VI) and let t_2 denote the number of FRI used for evaluation of witness polynomials (as described in section V).

1) Fri is used for retrieving the values of $S_{\sigma_1}(x)$, $S_{\sigma_2}(x)$, $S_{\sigma_3}(x), \mathbf{q_L}(\mathbf{x})$, $\mathbf{q_R}(\mathbf{x})$, $\mathbf{q_M}(\mathbf{x})$, $\mathbf{q_O}(\mathbf{x})$, $\mathbf{q_C}(\mathbf{x})$ at point $z \notin H$ sent by the verifier during one of the last steps of the protocol, hence $t_1 = 8$. Note that there is no need for using FRI for evaluation of any of $L_1(x), L_n(x), Z(x), S_{id_1}$, S_{id_2}, S_{id_3} as the polynomials are of very special reduced form and can be evaluated on behalf a verifier himself without any help from the prover. More precisely, polynomials S_{id_j} for $j \in [3]$ are linear, $L_i(x)$ for $i \in [n+1]$ are of the form:

$$L_(x) = \frac{c_i(x^{n+1}-1)}{x - g^i}$$

for some constant c_i and $Z(x)$ is of the form:

$$Z(x) = \prod_{a \in H^*}(x-a) = \frac{x^n - 1}{x-1}.$$

2) Witness polynomials $f_1(x) = \mathbf{f_L}(\mathbf{x})$, $f_2(x) = \mathbf{f_R}(\mathbf{x})$, $f_3(x) = \mathbf{f_O}(\mathbf{x}), T_0(x)$, $T_1(x), T_2(x), T_3(x)$ are evaluated at point z, Polynomials $P(x)$ and $Q(x)$ are evaluated at point z and $z \cdot g$ (recall, that we handle multi-evaluation with one instance of FRI). Hence $t_2 = 7$.

Assume that prover is malicious: he doesn't possess a valid witness and aims to cheat on a honest verifier. Let A denote the event for the prover to succeed in passing all verifier's checks. We will estimate the probability of event A. The probability is taken over random variables β, γ, z and all verifier's random variables sent during all FRI instances.

Assume that after interaction with prover verifier accepts the proof. The cases why it might happen fall into two major categories: either prover managed to cheat on verifier during any of FRI instances (we denote this event as B) or prover passed all FRI without cheating. Recall that soundness of any instance of FRI is denoted by $\mathsf{err}(\delta)$ Hence:

$$\Pr(A) = \Pr(A|B)\Pr(B) + \Pr(A|\bar{B})\Pr(\bar{B}) \leq$$
$$1 - (1 - \mathsf{err}(\delta))^{t_1 + t_2} + \Pr(A \cap \bar{B}).$$

We now restrict ourselves to the case of event $A \cap \bar{B}$ to happen. Due to soundness of *list commitments scheme* and *evaluation scheme*, event $A \cap \bar{B}$ means that all opening of setup polynomials are correct, and for every witness oracle $\hat{f}$ there exists a polynomial $f(x)$, such that $f(x)$ is δ-close to $\hat{f}$ and the opening provided by the prover is evaluation of $f(x)$ at corresponding point. Let L - be the maximal size of all the $|L_\delta|$ running over all witnesses' δ-lists. Note that L is bounded above by Johnson-bound $J_{\rho,\epsilon}$.

For prover to pass the protocol the equation $(*)$ should be satisfied at randomly chosen point z. There are at most L^{t_2} choices of substituted values in equation $(*)$ dependent on particular choices of functions $f_1, f_2, \ldots, f_{t_2}$ from corresponding δ-lists of oracles $\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_{t_2}$.

Let C be event that for any choice of polynomials $f_1, \ldots, f_{t_2}$ from aforementioned δ-lists the LHS of equation $(*)$ is not divisible by $Z(x)$ in the ring $\mathbb{F}[X]$. This in particular implies that for any choice of polynomials $T_0(x)$, $T_1(x)$, $T_2(x)$, $T_3(x)$ on the RHS the equation $(*)$ is not satisfied identically. Note that LHS and RHS of $(*)$ are both polynomials of degree at most $4n$ and if they are not equals they may coincide on at most $4n$ points (by Schwartz-Zippel lemma). Then:

$$\Pr(A \cap \bar{B}) = \Pr(A \cap \bar{B}|C)\Pr(C) + \Pr(A \cap \bar{B} \cap \bar{C})$$
$$\leq L^{t_2}\frac{4n}{|\mathbb{F}/D|} + \Pr(A \cap \bar{B} \cap \bar{C}).$$

Event $A \cap \bar{B} \cap \bar{C}$ means that for *some* choice of polynomials $P = \{f_1, \ldots, f_{t_2}\}$ equation $(*)$ holds identically. From now on we fix this set of polynomials P and show that with high probability they form a valid witness. Note that the set P may be efficiently derived by prover from corresponding oracles with the help of Sudan list-decoding algorithm, efficiently delivering proof of knowledge property.

For $P = \{f_1, \ldots, f_n\}$ and any of $F_1(x)F_2(x), \ldots, F_6(x)$ (where we substitute polynomials from P at appropriate places in $F_i(x)$), denote by D the event that this is not divisible by $Z(x)$. Using Lemma 1 we get:

$$\Pr(A \cap \bar{B} \cap \bar{C}) = \Pr(A \cap \bar{B} \cap \bar{C}|D)\Pr(D) + \Pr(A \cap \bar{B} \cap \bar{C} \cap \bar{D})$$
$$\leq \Pr(Z(x) \text{ divides } F(x) \mid \exists\, F_i(x) : Z(x) \nmid F(x)) + \Pr(A \cap \bar{B} \cap \bar{C} \cap \bar{D})$$
$$\leq \frac{1}{|\mathbb{F}|} + \Pr(A \cap \bar{B} \cap \bar{C} \cap \bar{D}).$$

Event $\bar{D}$ in particular means that $F_6(x)$ is identically zero on H^* and therefore $f_1(x), f_2(x), f_3(x)$, satisfy the second condition of the definition of witness for $\mathcal{L}'|_x$.

From $F_1(x), F_2(x), \ldots, F_5(x)$ being identically zero on H^* we retrieve (mimicking the reasoning in the proof of completeness property) that:

$$\prod_{i=1}^{n}\prod_{j=1}^{3}\Big(f_{(j-1)n+i} + \beta \cdot \tau\big((j-1)n+i\big) + \gamma\Big) = \prod_{i=1}^{n}\prod_{j=1}^{3}\Big(f_{(j-1)n+i} + \beta \cdot \tau \circ \sigma\big((j-1)n+i\big) + \gamma\Big) \tag{$\bullet$}$$

However, we have started with the assumption A that prover doesn't posses any valid witness for $\mathcal{L}'|_x$ and hence, as $f_1(x), f_2(x), \ldots, f_n(x)$, satisfy the second condition of the definition of witness, they can't satisfy the first (or they will constitute the witness in question). Using lemma 2 we then have:

$$\Pr(A \cap \bar{B} \cap \bar{C} \cap \bar{D})$$
$$\leq \Pr(eq. (\bullet) \text{ holds } \mid f_1(x), f_2(x), f_3(x) \text{ don't copy-satisfy } \sigma)$$
$$\leq \frac{1}{|\mathbb{F}|}.$$

Resuming all the computations we get, that the probability of malicious prover to cheat on a honest verifier is at most:

$$\Pr(A) \leq 1 - (1 - \mathsf{err}(\delta))^{t_1+t_2} + L^{t_2}\frac{4n}{|\mathbb{F}/D|} + \frac{2}{|\mathbb{F}|},$$

which is negligible (for precise numeric estimates of this probability refer to section IX). Moreover, we have shown that if prover manages to pass all the protocol's checks than any actual witness may be reconstructed (in polynomial number of steps) with the help of Sudan list decoding algorithm. This conclusion completes the soundness property proof.

C. Zero-knowledge

We will start with the explanation of the idea underlying zk-property proof. Presence of zero knowledge property means the existence of a simulator S not possessing any valid witness for $\mathcal{L}'$ such that S is able to generate a transcript $\langle S \rangle$ which is from side view indistinguishable from honest prover-verifier interaction $\langle P, V \rangle$ for any adversary D. Indeed, this is what zk-property is roughly saying: if there is no difference between transcript of prover-verifier interaction and simulator's output then prover's answers do not expose any information on the hidden witness (or the transcripts *should be* in some sort different). Recall that transcript in this context means a set of random variables $\langle a_1, a_2, \dots, a_n \rangle$, where a_i represent either verifier's messages and queries or prover's responses to corresponding queries. Note that transcript doesn't capture any information about the oracles themselves as we treat all oracles as being ideal and hence exposing no data except for the elements sent in response to verifier's queries to oracles (and those responses are encoded inside a_i). Note also, that in our case we have a public-coin protocol in which all verifier's queries are randomly distributed field elements.

We are going to construct the simulator S and transcript $\langle S \rangle$ in the following way: we will put as many variables of the transcript as possible being uniformly and randomly distributed. All remaining values will be uniquely fixed by the choice of previous random variables. We will then show that such an approach finally results in the requirement for witness polynomials f_i to have uniformly and randomly distributed values over some domains K_i (K_i are in general different for every witness polynomial). Then we will show that adding to each witness polynomial a masking polynomial $h_i(x)$ of degree at least $|K_i|$ is enough to achieve the required uniform distribution of values over K_i finishing the proof of ZK-property. (In reality we need to add masking of the form $z(x)h_i(x)$ as we don't want to change the values of any $f_i(x)$ on domain Z).

Now we are going to plug in all the details. The transcript of RedShift is the following: $\langle \beta, \gamma, z, \mathcal{T}\langle f_1 \rangle, \mathcal{T}\langle f_2 \rangle, \dots, \mathcal{T}\langle Q(x) \rangle$ (the range here is over all witness polynomials). where $\mathcal{T}\langle f \rangle$ denote the part of the transcript corresponding to relaxed polynomial commitment w.r.t. to witness oracle f. Note that we do not list the transcripts corresponding to the instances of polynomial evaluation protocol as it is conducted w.r.t to setup polynomials and those polynomials are considered to be public - we do not care how much information evaluation protocol exposes about them.

We start constructing the transcript $\langle S \rangle$ of S in the following way: we take β, γ to be uniformly randomly distributed over $\mathbb{F}$ and y to be uniformly and randomly distributed over $\mathbb{F}/H$. As those values are also taken at random on exactly the same domains by a honest verifier during the actual interaction with prover then this part of transcripts in $\langle S \rangle$ and $\langle P, V \rangle$ is totally equal. For $\langle S \rangle$ we also take the openings of each witness function except for $T_0(x)$ to be uniformly randomly distributed over $\mathbb{F}$. The evaluation of $T_0(x)$ at y is uniquely determined by equation $(*)$ which holds for any true transcript $\langle P, V \rangle$ and hence the same relation between variables should hold for simulator's transcript $\langle S \rangle$ for them to be indistinguishable (It can be easily observed that there are no other dependencies between the openings). Note, how we used our convention for $y \notin H$ here: in this case $Z(y) \neq 0$ and so we can obtain unique value for RHS of $(*)$ that will satisfy $(*)$ for any random choice of evaluations on LHS.

Now we are going to analyze transcript $\mathcal{T}(f)$ for any witness polynomial f in more detail. In the actual interaction between prover and verifier the transcripts $\mathcal{T}\langle f \rangle$ is of the following form:

$$i_1, i_2, \dots, i_k$$
$$z_1, z_2, \dots, z_k$$
$$x^{(0)}, x^{(1)}, \dots, x^{(r-1)}$$
$$a_0, s^{(0)}$$
$$q^{(0)}(s^{(0)}),\ q^{(0)}(t^{(0)}),\ \dots,\ q^{(r-1)}(s^{(r-1)}),\ q^{(r-1)}(t^{(r-1)})$$

where:

1) $i_1, i_2, \dots, i_k \in \mathbb{F}$ - the points in which verifier ask to open the oracle f. k is one for single-point evaluation (conducted for witness polynomials $f_1(x), f_2(x), f_3(x), T_0(x), T_1(x), T_2(x), T_3(x)$ at y: $i = i_1 = y$) and $k = 2$ for double evaluation (conducted for $P(x)$ and $Q(x)$ at points y and gy: $i_1 = y$, $i_2 = gy$).
2) $z_1, z_2, \dots, z_k$ - corresponding openings sent by the prover.
3) $x^{(0)}, x^{(1)}, \dots, x^{(r-1)}$ are random elements of $\mathbb{F}$ sent by verifier during the *commit* phase of FRI (which is, according to relaxed commitment protocol is conducted with respect to quotient function $q(x) = q^{(0)}(x) = \frac{f(x)-U(x)}{\prod_{l=1}^{k}(x-i_l)}$).
4) a_0 is the only coefficient of $f^{(r)} \in \mathbb{F}$ sent by prover at the end of the *commit* phase of FRI (note, that according

to the remark at the end of FRI section we assume all our instantiations of FRI to be fully unrolled and hence $f^{(r)}(x)$ to be constant. The proof of zero-knowledge property for the general case $deg(f^{(r)}) > 0$ is only a little harder and handled in a similar fashion).

5) $s^{(0)} \in D$ is the value chosen by verifier at the beginning of the *query* phase of FRI.
6) Every $s^{(i+1)} = q(s^{(i)})$ (for the definition of $q(x)$ refer to FRI section). $s^{(i)}, t^{(i)}$ is the coset of $s^{(i+1)}$.

The simulated transcript $\langle S\rangle$ of polynomial commitment of f is constructed in the following way:

1) The point (i) or two points (i_1, i_2) are already fixed by the previous history of $< S >$: $i = y$ or $(i_1, i_2) = (y, gy)$.
2) Similar for corresponding evaluations $z_1(z_1, z_2)$: recall, that they were are either chosen at random (for all witness polynomials except for $T_0(x)$) or defined uniquely by all previous values (for $T_0(x)$).
3) The values $x^{(i)}$ are distributed uniformly over $\mathbb{F}$ for honest verifier V. We take the same approach to simulator S: in $\langle S\rangle$ every $x^{(i)}$ is chosen uniformly at random over $\mathbb{F}$.
4) For S we take $s^{(0)}$ uniformly at random over $D = D^{(0)}$.
5) In $\langle S\rangle$ the values of $q^{(0)}(s^0)$ and $q^{(0)}(t^{(0)})$ are also taken uniformly at random over $\mathbb{F}$.
6) Recall that in true FRI protocol we have:
$$q^{(i+1)}(s^{(i+1)}) = p^{(i)}_{s^{(i+1)}}(x^{(i)})$$
where
$$p^{(i)}_{s^{(i)}}(x) := \text{interpolant}^{q^{(i)}|_{\{s^{(i)},t^{(i)}\}}}(x),$$
hence the value of every $q^{(i+1)}(s^{(i)})$ is uniquely determined by values $q^{(i)}(s^{(i)})$ and $q^{(i)}(t^{(i)})$ chosen on the previous "level" of FRI. In our simulator transcript this relation between values of $q^{(i)}(s^{(i)}), q^{(i)}(t^{(i)}), q^{(i+1)}(s^{(i+1)})$ should remain unchanged.
At step 5 we have fixed the values of $q^{(0)}(s^{(0)})$, $q^{(0)}(t^{(0)})$. From what said the value of $q^{(1)}(s^{(1)})$ in $< S >$ is then uniquely determined.
7) Now we try to simulate the value of $q^{(2)}(s^{(2)})$. As in previous paragraph, $q^{(2)}(s^{(2)})$ is uniquely determined by the values of $q^{(1)}(s^{(1)})$ and $q^{(1)}(t^{(1)})$ but for now only $q^{(1)}(s^{(1)})$ is fixed in $\langle S\rangle$ and $q^{(1)}(t^{(1)})$ remains undetermined. We lay $q^{(1)}(t^{(1)})$ uniformly at random over $\mathbb{F}$ and this will fix $q^{(2)}(s^{(2)})$ as well.
8) We proceed by induction: to fix the value $q^{(2)}(s^{(2)})$ we choose the value $q^{(2)}(t^{(2)})$ to be uniformly randomly distributed over $\mathbb{F}$. We then take the same approach for all downstream layers of FRI up to the bottom where we will eventually fix a_0 (which is determined by $q^{(r-1)}(s^{(r-1)})$ and $q^{(r-1)}(t^{(r-1)})$. This completes the construction of simulation $\langle S\rangle$.

Our next aim is to achieve the same distribution for transcripts in honest prover-verifier interaction $\langle P, V\rangle$. First, for all witness polynomials (except for T_0, but it is not of big importance for now) we want their values at z to look like randomly distributed values over $\mathbb{F}$. This means, that all these polynomials should have one additional "degree of freedom": allow the prover to replace every witness $f_i(x)$ by $f'_i(x) = f_i(x) + aZ(x)$, where $a \in \mathbb{F}$ is some constant chosen by prover uniformly random. This modification will not violate the protocol (as we are actually interested in the values of witness polynomials on domain H), but let the prover to achieve the desired property for uniform distribution of $f'(z)$.

There are more uniformly distributed random variables that come from our construction of simulator S for FRI subprotocol. Recall that due to our construction of $< S >$ we want the values
$$q^{(0)}(s^{(0)}), q^{(0)}(t^{(0)}), q^{(1)}(t^{(1)}), q^{(2)}(t^{(2)}), \dots, q^{(r-1)}(t^{(r-1)})$$
to look like being completely uniformly random. In order to achieve this we need the following lemma:

Lemma 4. *Let $f(x)$ be interpolation polynomial of $Z = \{z_1, z_2, \dots, z_n\}$ over domain $I = \{i_1, \dots, i_n\}$ (which means that $f(x)$ is the unique polynomial of degree $\le n-1$, such that $f(i_k) = z_k$, for all $k \in [n]$). Let x be any point in $\mathbb{F}$ different from all $i_2, \dots i_n$. If z_1 runs uniformly over all of $\mathbb{F}$ then $f(x)$ also runs uniformly over $\mathbb{F}$.*

Proof. Recall the definition of Lagrange interpolation polynomial:
$$f(x) = \sum_{j=1}^{n} \prod_{k \neq j} \frac{x - i_k}{i_j - i_k} z_j.$$
With fixed $x, i_1, \dots, i_n$ and $z_1, \dots, z_n$ $f(x)$ is a function of z_1 of the form:
$$f(x) = az_1 + b.$$
where a, b - constants $\in \mathbb{F}$. Note that
$$a = \prod_{k \neq 1} \frac{x - i_k}{i_1 - i_k} \neq 0,$$
provided x being different from all of $i_2, \dots i_n$. Now, for linear function the conclusion of the lemma is obvious. □

Consider $s^{(01)}$ and $t^{(01)}$ - the coset of $t^{(1)}$. Although the evaluations at these points are never explicitly shown in the transcript they are of severe importance of us. Indeed, at least one of $s^{(01)}$ or $t^{(01)}$ is unequal to $x^{(0)}$. Without loss of generality assume $t^{(01)} \neq x^{(0)}$. We use lemma 4 for $I = \{s^{(01)}, t^{(01)}\}$, $Z = \{q^{(0)}(s^{(01)}), q^{(0)}(t^{(01)})\}$ and $x = x^{(0)}$ which implies that uniform distribution of $q^{(0)}(t^{(01)})$ results in uniform distribution of $q^{(1)}(t^{(1)})$, independent of value $q^{(0)}(s^{(01)})$.

We proceed by induction with repetitious use of lemma 4. To achieve uniformly random distribution of $q^{(2)}(t^{(2)})$ we need uniformly random distribution of one of values from the previous level: $q^{(1)}(s^{(12)})$ or $q^{(1)}(t^{(12)})$. Assume that $s^{(12)} \neq x^{(1)}$ and hence satisfy the conditions of lemma 4. The uniform distribution of $q^{(2)}(t^{(2)})$ then follows from uniformly random distribution of one of $q^{(1)}(s^{(12)})$ which in turn follows

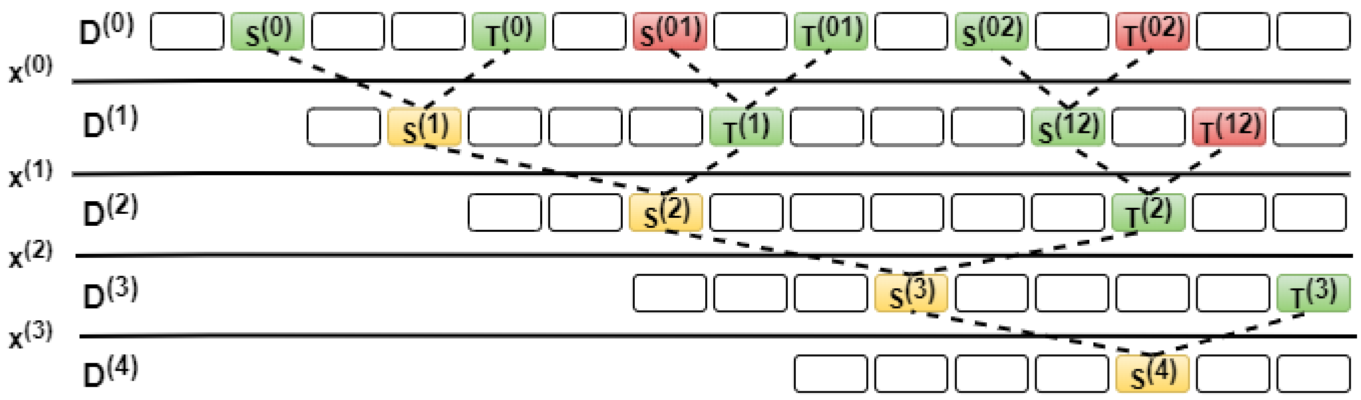

Fig. 2: FRI transcript

from uniform distribution of (say) $q^{(0)}(s^{(02)})$. The same logic is then applied for all downstream layers of FRI.

To aid your intuition with what happens in the previous paragraph, consider fig. 2 depicting the first view layers of FRI.

Here bold lines separate adjacent levels of FRI, green blocks determine the values that are taken uniformly at random, yellow blocks represent the values that are uniquely determined by the values of corresponding coset on the previous layer and red blocks have no impact in the constructions.

To sum up, to achieve the same distribution of variables in transcripts $\langle P, V\rangle$ and $\langle S\rangle$ we need to add more "degrees of freedom" for each witness polynomial. More precisely, we want the evaluation

$$q^{(0)}(s^{(0)}),\ q^{(0)}(t^{(0)}),\ q^{(0)}(t^{(01)}),$$
$$q^{(0)}(s^{(02)}),\ \dots\ (r+1 \text{ values in total})$$

over the set $K' = \{s^{(0)}, t^{(0)}, t^{(01)}, s^{(02)}, \dots\}$ on the top level of FRI to be uniformly and randomly distributed.

Now, recall that:

$$q^{(0)}(x) = q(x) = \frac{f(x) - U(x)}{\prod_{l=1}^{k}(x - i_l)} \quad (**)$$

.

Recall that in our case the sets $\{i_1, \dots, i_k\}$ and D are disjoint. This in turn means that uniformly random distribution of values of $q^{(0)}(x)$ over K' is exactly the same as the uniformly random distribution of values of f over the same domain (as all other terms in $(**)$ are now fixed by previous considerations). Plugging in the the requirement for $f(z)$ to be also uniformly randomly distributed we arrive at the set $K = K' \cup z$ with $|K| = r+2$ at which the values of f should look like random elements in $\mathbb{F}$. To achieve this property, it is enough to replace $f(x)$ by $f'(x) = f(x) + h(x)Z(x)$ where $h(x)$ is any polynomials of degree $r+1$ (note, that we use the fact the sets K and H are disjoint).

This completes the proof of zero-knowledge property and the whole analysis of RedShift.

Appendix C
FRI parameters

As described in the main text of the paper and in particular in Section IX, one has the freedom to pick FRI parameters that also affect contributions into the soundness error of RedShift due to the list size $|L|$. In general, *smaller* list sizes will lead to a *smaller* δ parameter (this is intuitively expected, as a larger list size requires less sensitivity) that in turn *reduces* FRI soundness for a chosen domain D, parameter ρ and number of queries. Alternatively, one can pick another limit in the FRI soundness formula

$$p = \Big(1 - min\Big\{\delta_0, J_\epsilon(J_\epsilon(1-\rho))\Big\} + \epsilon \log|D|\Big)$$

and set $\delta_0 = \frac{1-\rho}{2}$ to be in the unique decoding radius. In this case list size $|L| = 1$, but FRI has *smaller* soundness for the same number of queries. This means that one has to pay particular attention to the final system soundness as described in Section X: for in case where the sum $A_{SZ} + A_{FRI} \sim A_{SZ}$ one should also consider the case of the limit $\delta_0 = \frac{1-\rho}{2}$ in the FRI soundness term and can recalculate A_{SZ} and thus a final soundness in case of list size $|L| = 1$.

Such checks are also important if one would want to reduce the field size for a corresponding reduction in proof size.

Appendix D
Proof size and optimizations

There are various options that can reduce the proof size. Some of the are described in detail in [37]. There are two essential parts to check satisfiability at the random point:

1) Consistency between openings of various polynomials at the random point i. The prover sends the purported evaluations to the verifier and these values are used for two subroutines:
   a) Check the equations from Section VII at this random point.
   b) Perform simulation of virtual oracles to the quotient functions in the form $q(x) = \frac{g(x)-g(i)}{x-i}$ that are later used as an input to the FRI protocol.

2) Proximity testing performed by invocation of the FRI protocol

As described in [37], the prover can join the evaluations of various polynomials g_k at the domain D into a single oracle by placing the corresponding values into the same leaf of the Merkle tree thus reducing total number of Merkle paths required for authentication.

We can perform such a joining operation for the following sets of polynomials:

1) Constraint polynomials: selectors q_L, q_R, q_O, q_M, q_C and permutation polynomials $S_{id_1}, S_{\sigma_1}, S_{\sigma_2}, S_{\sigma_3}$ as all those are independent and prepared simultaneously at the setup step.
2) Witness polynomials f_L, f_R, f_O.
3) Grand product polynomials P, Q.
4) Polynomials $T_{low}, T_{mid}, T_{high}$.

While initially we would have to provide 17 independent Merkle paths for authentication of various oracle values now we can reduce this number to 4.

Next we can place more values (2^k) into leafs of every Merkle tree used to instantiate the oracles. This optimization allows the verifier to get more values from the oracle for the small change of total communicated data size. This also allows one to perform k interpolation steps before accessing values from the next subdomain oracle in FRI. We do not analyze whether such an operation is equivalent to testing 2^{k-1} independent starting values to FRI and thus effectively improves soundness by the factor of 2^{k-1} per query. Note that we still use a localization factor of 2 for FRI even while more values are placed in the leaf.

There also exist other optimization e.g. to perform proof-of-work on top of challenge values obtained from the transcript to reduce number of required queries for FRI protocol (as used in [38]) and other estimates for the number of required queries. To the best of our knowledge there is no public analysis of such optimizations and we will not use them in our analysis. For a rate of $\rho = 1/16$ we place 8 values per leaf in Merkle tree, which allows us to perform 3 interpolations simultaneously in FRI - thus saving on Merkle paths. Thus our final proof size for a case of $n+1 = d = 2^{20}$ and domain size of $|D| = d/\rho = 2^{24}$ will have the following contributions:

1) 4 Merkle roots for 4 aggregated oracles to polynomials enumerated above.
2) Queries to the oracles to 4 polynomials enumerated above on the domain D: $8 + log(2^{24}) - log(8) = 29$ of 32 byte values per query per aggregated polynomial oracle.
3) Merkle roots to FRI intermediate oracles. There are oracles for domains of sizes $21, 18, \ldots, 6, 3$ (keep in mind that we use 8 values per leaf) for a total of 7.
4) Final interpolation coefficients for FRI: 1 coefficient (FRI is fully unrolled and it may be not the optimal solution).
5) Queries to the final batched FRI over quotients: as described above we assume testing of 4 cosets simultaneously per query that is ~ 4 bits of security for a rate $\rho = 1/16$. One will need to communicate $8-1$ elements per intermediate FRI domain (one element is calculated from interpolation steps) and Merkle paths for each of the oracles for domains of sizes $21, 18, \ldots, 6, 3$, (7 domains) in total of $7 \cdot 7 + (21 + 18 + \cdots + 6 + 3) = 133$ of 32 byte values per query.

Thus we can calculate a total proof size for 80 bits of security: we will need 80 queries with a final consisting of $4 + 1 + 7 + 80 \cdot (4 \cdot 29 + 133) = 19932$ of 32 byte values in a total 623 KB (1 KB = 1024 bytes). For comparison if we do not implement optimizations and use only naive approaches at every step we should use 80 queries and 17 independent oracles to polynomial evaluations on D then proof size would be $17 + 1 + 23 + 80 \cdot (17 \cdot 24 + 24 + 23 + \cdots + 2 + 1) = 54761$ of 32 byte values in a total 1772 KB.

Not all the optimizations were implemented in the PoC prover as of the date of writing, but effect of optimizations is expected to also slightly benefit proving times.

For illustrative purposes we also demonstrate proof sizes if *conjecture* from [28] is correct. In this case number of queries can be calculated as $\sim -\lambda/log(\rho)$ where λ is our security parameter. We target again 80 bits of security with the following proof sizes if conjecture holds as shown in a table II.

TABLE II: Expected proof sizes for various setup parameters if conjecture from [28] is valid.

Target security level, bits	Rate	Problem size	Proof size, optimized, under conjecture from [28]
80	1/16	2^{20}	156 KB
80	1/16	2^{24}	191 KB
80	1/16	2^{28}	234 KB
80	1/32	2^{20}	131 KB
80	1/64	2^{20}	123 KB
80	1/128	2^{20}	110 KB
80	1/256	2^{20}	96 KB
120	1/16	2^{20}	234 KB
120	1/16	2^{24}	286 KB
120	1/16	2^{28}	351 KB