Vortex: A List Polynomial Commitment and its Application to Arguments of Knowledge

Alexandre Belling, Azam Soleimanian, and Bogdan Ursu

Linea, Prover Team
{firstname.lastname}@consensys.net

Abstract. A list polynomial commitment scheme (LPC) is a polynomial commitment scheme with a relaxed binding property. Namely, in an LPC setting, a commitment to a function $f(X)$ can be opened to a list of low-degree polynomials close to $f(X)$ (w.r.t. the relative Hamming distance and over a domain D). The scheme also allows opening one of the polynomials of the list at an arbitrary point x and convincing a verifier that one of the polynomials in the list evaluates to the purported value.
Vortex is a list polynomial commitment, obtained through a modification of Ligero (CCS 2017), inspired by the schemes of Brakedown (Crypto 2023), batch-FRI (FOCS 2020), and RedShift (CCS 2022). Concerning one application of Vortex, for a witness of size N, the messages between the prover and the verifier are of size $O(N^{1/2})$. Vortex is a core component of the SNARK used by the prover of Linea (Consensys). This paper provides a complete security analysis for Vortex. We use a general compiler to build an Argument of Knowledge (AoK) by combining our list polynomial commitment and a polynomial-IOP (PIOP).
The approach is similar to combining a PIOP with a polynomial commitment scheme and has a soundness loss only linear in the list size. This overcomes a previous limitation in the standard compiler from a generic PIOP and a list polynomial commitment scheme to an interactive argument of knowledge, which suffers from a soundness loss of $\mathcal{O}(|L|^r)$ (where $|L|$ is the list size and r is the number of interactions between the prover and the verifier in the PIOP).

Keywords: List Decoding Regime, Polynomial Commitment, List Polynomial Commitment, Reed-Solomon Codes, SNARK, PIOP, Linea.

1 Introduction

Polynomial Commitments A polynomial commitment scheme (PCS) [24] is a cryptographic primitive in which a prover commits to a polynomial $P(X)$ and later proves the evaluation of $P(X)$ at a given point x.

List Polynomial Commitments (LPC) An LPC is a polynomial commitment with a relaxed security requirement: the commitment is not associated with a single polynomial but with a list of polynomials, where the prover can open the commitment to any polynomial from the list. In other words, the commitment is not binding to one polynomial but to a list of polynomials.

Succinct Non-Interactive Arguments of Knowledge (SNARKs) Given a binary relation $\mathcal{R}(x, w)$, SNARKs allow proving knowledge of a witness w such that the relation $\mathcal{R}$ (usually drawn from a large family) is satisfied for a public input x. In particular, the verifier needs less time to verify the proof, generated by the SNARK, rather than to perform again all the computations. In the last few years, an ever-growing number of SNARK constructions have emerged, including Groth16 [20], Plonk [4], Halo [13], Halo2 [17], Marlin [16], Spartan [28], Virgo [35], Brakedown [19], Orion [33], Libra [32], Aurora [7], Fractal [15], Sonic [26], Nova [23], and Lasso [29] to cite a fraction of the existing works.

zk-VMs and zk-EVMs In a state machine, a transition is the process of moving from an old state to a new state by reading a series of inputs and performing sets of opcodes which are a limited and low-level set of instructions. Ethereum is, in essence, a transaction-based state machine, where the state contains all account addresses and their mapped account states. The Ethereum Virtual Machine (EVM) is the mechanism

responsible for performing the transitions as a succession of opcodes. zk-VMs (zk-Virtual Machines) and, more specifically, zk-EVM (Ethereum Virtual Machine) are complex cryptographic systems that allow one party to generate proofs assessing the correct execution of a Virtual Machine using a SNARK scheme[1]. The proofs can be as short as a few hundred bytes and be verified in a few milliseconds on any platform (Groth16 [20]). For these reasons, zk-VMs have important applications in blockchain scalability and interoperability and have seen tremendous activty in research and development: Linea [5], Cairo [18], Polygon-zkEVM [30], RISC0 [34], Scroll [1]. However, building a system capable of proving arbitrary executions of the Ethereum Virtual Machine is no easy task. At a high level, the zk-EVM of Consensys [5] models execution traces of the Ethereum Virtual Machine using hundreds of polynomials and thousands of arithmetic constraints of various types. In this setting, the total witness size for proving the execution of a regular block consists of hundreds of millions of field elements.

Interactive Oracle Proofs Interactive Oracle Proofs (IOP) are a family of abstract ideal protocols in which the verifier is not required to read the prover's messages in full. Instead, the verifier has oracle access to the prover's messages and may probabilistically query them at any positions [6]. IOP protocols can be transformed into concrete secure argument systems using a Merkle tree. Later works have introduced several variants of IOP such as polynomial-IOP (PIOP) or tensor-IOP, where the prover can perform polynomial evaluation queries [4] or tensor queries [12]. Similarly, these protocols can be converted into concrete argument systems (including SNARK) using functional commitments. This type of approach for building argument systems has now become a standard [4].

Reed-Solomon Encoding and Decoding Regimes Generally speaking, the Reed-Solomon encoding receives the evaluations of a function over k points, considers them as the coefficients (or evaluations) of a polynomial $P(X)$, and then outputs the evaluation of such a $P(X)$ over a fixed set D (usually the set of roots of unity over the finite field $\mathbb{F}_q$). The output is called a codeword of size $|D|$.

Considering the relative Hamming distance as the measure, a decoding algorithm receives the vector w over D and outputs codewords close to w. For the Reed-Solomon code, one considers the unique decoding regime and list decoding regime. In the unique decoding regime, the radius of the ball around w (w.r.t. the relative Hamming distance) is small and there is only one codeword that can be that close to w, while in the list decoding regime the radius is bigger and there are many codewords that fall in the ball around w.

1.1 Our Contributions and Techniques

Here we summarize our contributions.

A Compilation Framework from PIOP to UniEval PIOP We introduce a compiler (adapted from [11]) that allows transforming any secure PIOP into one where the verifier sends oracles queries on a single opening point for all polynomials (while it may have sent queries directly to the prover before this step). As the original goal of our work is to build a succinct proof system for the zk-EVM specified by Linea [5], this compiler approach has numerous benefits. An important one is that it allows specifying and implementing batching and optimization techniques that would be a lot more complex otherwise. The main feature of this compilation is that it yields a single-point evaluation PIOP, allowing us to use it alongside a non-homomorphic polynomial commitment (i.e., Vortex) to create an efficient argument system. Indeed since Vortex is not a homomorphic commitment scheme, the well-known batching techniques based on random linear combinations are not readily applicable, but with UniEval PIOP we only require the possibility of batching over the same point. This batching feature is supported by Vortex.

Vortex, a Batchable Polynomial Commitment (BPC) A polynomial commitment allows a prover to open the committed polynomial over a given point. A Batchable Polynomial Commitment (BPC) allows the same type of opening for a batch of committed polynomials on the same point.

[1] In the blockchain community, sometimes the term zk is employed to mean succinctness.

In Section 5, we present Vortex, an adaptation of Ligero [3] into an BPC scheme inspired by the works of Brakedown [19], batch-FRI [10], and RedShift [22].

As Brakedown, our BPC does not rely on FRI protocol and it has a proximity check and an evaluation check where the proximity check is indeed the Ligero test. The main difference from Brakedown is the security regime we are dealing with. Based on encoding schemes, one can imagine two security regimes: the unique decoding regime that is the counterpart for the standard binding and the list decoding regime leading to a relaxed binding property where the commitment can be opened to a fixed list.

Working in the list decoding regime requires a new design. Indeed, the evaluation protocol of Vortex is different from the one in Brakedown, where we combine the proximity check and evaluation check as the evaluation protocol. More precisely, in Brakedown, the proximity check can be run independently of the evaluation point, while in Vortex the proximity check is run after seeing the evaluation point.

We show that a polynomial commitment scheme in the list decoding regime (Vortex LPC) is enough for the compilation of PIOP to an argument of knowledge (AoK).

From the instantiation point of view, for hashing the columns, our Vortex scheme relies on a hash function based on the Ring-SIS assumption [25] where we also apply an MIMC hash over the output of the SIS-hash. The first instance of Ring-SIS-based hash functions was introduced in [25]. It is a SNARK-friendly hash function with a linear structure defined over the ring of polynomials of degree less than d, as $H_a(s) = \sum a_i(x)s_i(x) \in \mathcal{R}$ for $\mathcal{R} = \mathbb{Z}_q(X)/X^d + 1$. Another advantage of using such a hash function is the possibility of using lookup arguments if the hash computation is not done on the verifier side. To encode the rows, we use (systematic) Reed-Solomon encoding [27].

Compilation of a PIOP to an Argument of Knowledge (AoK) in the List Decoding Regime As mentioned, we discuss the security of Vortex in the list decoding regime. That can only guarantee a relaxed binding property and is not enough for a standard compilation of PIOP to AoK. We thus present a framework to compile a PIOP to AOK via LPC. If such a compilation is done naively the soundness error can go up to a factor $|L|^r$ where $|L|$ is the list size in LPC and r is the number of rounds in PIOP. We present a compilation technique that increases the soundness error only by a factor $|L|$.

1.2 Overview of Vortex

Vortex As for Brakedown [19] and [33], the Vortex construction is simple and constitutes a modification of Ligero [3]. Assume that $\mathcal{P}$ and $\mathcal{V}$ are the prover and the verifier. First, we elaborate on the commitment procedure. The prover commits to a matrix W of size $m \times k$ as follows:

- **Row-Encoding:** $\mathcal{P}$ starts by encoding the matrix rows to obtain a new matrix W' of size $m \times n$.
- **Column-Hashing:** The prover then hashes each column of W' and sends them to the verifier as a commitment

For the evaluation, the verifier wishes to know if there exists a matrix-codeword G (whose extension over D is) close to W' such that $G \cdot l = y$ with $l = (1, x, \cdots, x^k)$. For this, first, they follow the Ligero test as follows:

Proximity Check:
- for some vector $\mathcal{B}$ drawn at random by the verifier, the prover sends a vector u.
- the verifier encodes u to u' using the Reed-Solomon code.
- it queries the openings of t random columns of W', where the prover sends the columns in response.
- For the opened columns, the verifier checks;
  1. whether the alleged column openings are consistent with the corresponding hash values from the commitment.
  2. whether the scalar-products of $\mathcal{B}$ and the chosen columns are consistent with u'.

  The test ensures that W' is close to a Matrix-codeword (i.e., a matrix whose rows are codewords). Namely, that there exists a list L of Matrix-codewords close to W'.

Evaluation Check: Finally, by checking $u \cdot l = \mathbf{B} \cdot y$, one of the matrices in L has to evaluate to y over l.

One difference from Brakedown is in the evaluation phase which allows us to support batching separate polynomials in the same committed matrix. More importantly, we analyze the security in the list-decoding regime and prove that Vortex is an LPC scheme. Working in this regime brings some subtlety to the compilation of PIOP to AoK. We also present a compiler compatible with LPC (rather than PC).

Our compiler has several important properties:

- it uses an LPC rather than a (binding) polynomial commitment.
- the soundness error increases only by a factor L (comparing to the compilation via polynomial commitment or the schemes in the unique decoding regime).

2 Related Works: Ligero-Based Polynomial Commitments

One can analyze the soundness of Vortex in two different regimes: "unique decoding regime" and "list decoding regime". Intuitively, in the unique decoding regime the commitment is guaranteed to be binding. In the list decoding, the commitment satisfies a relaxed binding property where the commitment can be opened to any polynomial close to the committed one. This technically means that the polynomial commitment is not knowledge-sound with respect to the standard relation. In [22], such a polynomial commitment scheme is called a list polynomial commitment (LPC).

In the list decoding regime (for Vortex), the efficiency improvement comes from the verifier opening fewer columns compared to the unique decoding regime.

Following the strategy in DEEP-FRI [8] and RedShift [22], we discuss the knowledge-soundness of Vortex in the list decoding regime.

Both Breakdown and Orion follow the unique decoding regime. Theoretically, to guarantee that binding holds, one just needs to set the distance on the unique decoding radius (the parameter setting of the unique decoding regime), but as already mentioned, this requires opening more matrix columns.

2.1 Interactive Oracle Proof (IOP) of Proximity

A Fast Reed-Solomon Interactive Oracle Proof (FRI) [9] is an IOP for testing the proximity to a codeword.

Namely, given a function $f : D \to \mathbb{F}$, the aim in FRI is to prove that f corresponds to a low-degree polynomial with respect to the size of D. The oracle provided by the FRI prover is the function f, and the verifier queries the values at points from D. Due to the small size of D (compared to the finite field $\mathbb{F}$), the key tool for distinguishing if f is a codeword is statistical sampling [21]. However, a statistical test can only ensure proximity, which we measure by the relative Hamming distance $\delta(f, g)$. Thus, in FRI the prover convinces the verifier that a given function $f : D \to \mathbb{F}$ is close (and not necessarily equal) to a low-degree polynomial, i.e. $\delta(f, P) \leq \theta$ for some polynomial $p(X)$ of specified maximum degree. In other words, f agrees with $p(X)$ on a set $A \subset D$ of density $|A|/|D| \geq 1 - \theta$. In applications, the agreement set A is chosen to be large enough to infer global properties on the low-degree polynomial.

2.2 Ligero Testing and the Correlated Agreement Theorem

The Ligero test [3] checks the proximity of a batch holding a known codeword. Consider the batch $\{f_i\}_{i\in[m]}$. In the Ligero test, the verifier sends a random $\lambda \leftarrow \mathbb{F}$, and the prover provides oracle access to the linear combination $f := \sum \lambda^i f_i$. The verifier then samples some columns and checks the consistency for the linear combination. Finally. the test ensures that each f_i is close to a codeword, "If f is close to a codeword".

The soundness of the Ligero test is argued based on the Correlated Agreement Theorem, which informally says that if the linear combination is at distance $\leq \theta$ from a codeword, then each f_i must follow a similar proximity property.

2.3 Brakedown versus batch-FRI.

The Proximity Check of the "batch" in Brakedown and Batch-FRI [10] is their common point which is indeed the Ligero test [3]. The difference is how the "if" statement of Ligero is satisfied (remember that "if" f is close to a codeword then each element of the batch is close to a codeword).

In batch-FRI, the verifier has oracle-access to each function f_i and to the linear combination over D (i.e., $f|_D$), while in Brakedown the verifier has oracle-access to the functions f_i over the domain D, and "full"-access to $f|_D$ where it actively participates in building f as a codeword (In Vortex, similarly to Brakedown and Orion, this is done by giving full access to u when the verifier expands it to u'. The vector u' is then the equivalent of f described above). From this point, Brakedown and batch-FRI act differently: batch-FRI applies FRI over the result of the linear combination $f|_D$ to check that f is close to a codeword, while in Brakedown (and also in Vortex), the verifier knows the codeword f. The fact that the verifier knows the codeword f makes Brakedown simpler and improves the proof-time, while it increases the proof size to k (the size of u).

Relying on the Ligero test in all these schemes (Brakedown, Orion, batch-FRI) the soundness is based on the Correlated Agreement Theorem. Brakedown and Orion follow the unique decoding regime, while batch-FRI works in different regimes. For Vortex, we present the soundness analysis in the list-decoding regime (which requires the opening of fewer random columns). The compilation of PIOP to AOK via Vortex is simple and efficient, albeit at the cost of losing a small factor in the soundness error. We borrow the idea of the DEEP query of Deep-ALI (this is what we call the Grail query in UniEval compilation) to improve the soundness of the compilation in the list-decoding regime while keeping the simplicity of Brakedown.

3 Preliminaries

Here we define the syntax of our main primitives: polynomial commitments, list polynomial commitments, IOPs and arguments of knowledge.

3.1 Interactive Protocols and Arguments of Knowledge (AoKs)

In this subsection, we roughly follow the definitions of [14]. An NP relation $\mathcal{R} \subseteq \{0,1\}^*$ is a set of binary-string tuples (x, w) for which there exists a polynomial-time algorithm that on input (x, w), determines whether $(x, w) \in \mathcal{R}$ (the polynomial runtime is w.r.t. $|x|$, $|w|$). The NP language $\mathcal{L}_\mathcal{R}$ associated to $\mathcal{R}$ is defined as: $\mathcal{L}_\mathcal{R} := \{x : \exists w \text{ s.t.} \mathcal{R}(x, w) = 1\}$. The string x is called the statement, and it is called valid when $x \in \mathcal{L}_\mathcal{R}$. The string w is the witness of x, allowing to efficiently check membership of x to the language.

Interactive Proofs of Knowledge Interactive proofs of knowledge are a particular case of two-party protocols between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$. Both parties have access to an NP statement x, and the prover is given access to a witness w for the statement. The protocol is complete if the interaction between $P(x, w)$ and verifier $V(x)$ always results in the verifier accepting.

Consider an arbitrary (potentially malicious) prover P^* for which the verifier $V(x)$ accepts with non-negligible probability. Knowledge-soundness is satisfied if there exists an extraction algorithm E (called the extractor) that given access to the arbitrary prover P^* will output a valid witness w with overwhelming probability. The extractor can inspect the internal state of P^*, run it for any number of steps and rewind it to any previous state.

We denote the random variable which constitutes the output of a verifier on a run of an interactive protocol on a statement x as $\langle P(x; r_P) \leftrightarrow V(x; (r_1 \dots r_n))\rangle$, where r_P is the randomness of the prover and $r_1 \dots r_n$ are the public coins which constitute the verifier messages in verifier rounds 1 to n. When we do not need to specify the exact random coins used, we resort to the shorthand notation $\langle P(x) \leftrightarrow V(x)\rangle$.

In cases where the relation R is defined w.r.t. parameters relevant to other protocols, we consider a PPT relation generator $\mathcal{R}_\lambda$, namely $\mathcal{R} \leftarrow \mathcal{R}_\lambda$.

Definition 1 (Interactive Arguments of Knowledge). *We consider interactive protocols in which both the prover and the verifier are provided access to a common reference string, generated in an offline setup phase (the reference string will be implicitly assumed to be given as input to all the following algorithms). Such an interactive protocol (P, V) between a prover P and a verifier V is an argument of knowledge for an* NP *relation $\mathcal{R}$ if it satisfies the following two properties:*

- *Perfect Completeness: for all $(x, w) \in \mathcal{R}$, it holds that:*

$$\Pr[\langle P(x, w) \leftrightarrow V(x)\rangle = 1] = 1.$$

- *Soundness: this property states that it is not feasible to convince the verifier of a wrong statement. More formally, for any non-uniform PPT adversarial $P' = (\mathcal{A}, P^*)$ we have,*

$$\Pr[\langle P^*(x) \leftrightarrow V(x)\rangle = 1 \ \wedge \ x \notin \mathcal{L}_{\mathcal{R}} : \ \mathcal{R} \leftarrow \mathcal{R}_\lambda, x \leftarrow \mathcal{A}(\mathcal{R})] \approx 0$$

- *Knowledge Soundness in the random oracle model: there exists an algorithm E that given oracle access to the adversarial prover P^* (along with access to the statement, all adversarial queries to the random oracle, the possibility to inspect the adversarial state, rewind and execute for a specified number of steps) satisfies that: $E^{P^*(x)}$ runs in probabilistic polynomial-time and moreover $\delta(|x|) = \mathrm{negl}(\lambda)$, where δ is defined as:*

$$\delta(|x|) = \Pr\left[\langle P(x) \leftrightarrow V(x)\rangle = 1 \ \wedge \mathcal{R}(x; w) = 0 : \begin{array}{c}\mathcal{R} \leftarrow \mathcal{R}_\lambda, \\ (x, w) \leftarrow E^{P^*}(\mathcal{R})\end{array}\right]$$

Succinctness Our focus is on succinct interactive protocols. Informally, both the prover and the verifier time, as well as the size of the messages sent between the two parties must be small compared with the witness of the relation being proven.

.

3.2 (Batched) Polynomial Commitments

The definitions in this section follow the presentation from [11]. Similarly to [11], we considered batched openings of multiple polynomials. One difference is that we only consider openings of all these polynomials at the same evaluation point.

Definition 2. *A (batched) polynomial commitment scheme (PCS) is a triplet* $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ *that satisfy:*

- $\mathsf{Setup}(1^\lambda, k)$ *generates public parameters* pp *(a structured reference string) suitable to commit to polynomials of degree $< k$.*
- $\mathsf{Commit}(\mathsf{pp}, P_1(X) \dots P_n(X))$ *outputs a commitment C to n polynomials $(P_1(X) \dots P_n(X))$ of degree $< k$ using* pp.
- $\mathsf{OpenEval}$ *is a (public-coin) protocol between two parties, a prover P_{PC} and a verifier V_{PC} that either accepts or rejects. The prover is given n polynomials $P_1(X) \dots P_n(X) \in \mathbb{F}_{<k}[X]$. Both parties receive the following:*
  - *security parameter λ, degree bound k and batch size n, such that $k, n = poly(\lambda)$.*
  - *The public parameters* pp*, where* $\mathsf{pp} = \mathsf{Setup}(1^\lambda, k)$.
  - *An evaluation point x and alleged openings $y = (y_1 \dots y_n)$.*
  - *Alleged commitment C for polynomials $P_1(X) \dots P_n(X)$.*

Definition 3 (Completeness of an Batched Polynomial Commitment Scheme). *We say that a batched polynomial commitment scheme has perfect completeness if for any security parameter λ, any integers $k, n = \mathrm{poly}(\lambda)$, any polynomials $P_1(X) \dots P_n(X) \in \mathbb{F}_{<k}[X]$, arbitrary evaluation point x and alleged opening y, if $C = \mathsf{Commit}(\mathsf{pp}, P_1(X) \dots P_n(X))$ and $P_i(x) = y_i$ for all $i \in [n]$ then an interaction of $(P_{\mathrm{PC}}, V_{\mathrm{PC}})$ where P_{PC} runs on the aforementioned parameters will result in the verifier accepting with probability one.*

Definition 4 (Knowledge Soundness in the Random Oracle Model). *There must exist a* PPT *extractor E such that for every* PPT *adversary $\mathcal{A}$ and arbitrary degree $k = \text{poly}(\lambda)$, the probability that $\mathcal{A}$ wins the following game is negligible, where the probability is taken over the coins of* Setup*, $\mathcal{A}$ and V_{PC}. Moreover, the extractor has access to the random oracle queries of $\mathcal{A}$:*

- *$\mathcal{A}$ receives degree k and $\mathsf{pp} = \mathsf{Setup}(1^\lambda, k)$. $\mathcal{A}$ outputs C.*
- *E receives the commitments C, inspects the random oracle queries made by A in the previous step and outputs $P_1(X) \dots P_n(X) \in \mathbb{F}_{<k}[X]$.*
- *$\mathcal{A}$ outputs an evaluation point x and claimed openings y.*
- *$\mathcal{A}$ interacts with the V_{PC} verifier of the* $\mathsf{OpenEval}$ *algorithm. The inputs of $\mathcal{A}$ for this subprotocol are $C_1 \dots C_n$, x and y.*
- *$\mathcal{A}$ succeeds if V_{PC} accepts but $P_i(x) \neq y_i$ for some $i \in [n]$.*

3.3 IOP and Polynomial-IOP

An interactive oracle proof (IOP) for a relation $\mathcal{R}(x, w)$ is an interactive proof in which the verifier is not required to read the prover's messages in their entirety; rather, the verifier has oracle access to the prover's messages, and may probabilistically query them. In Polynomial IOP, the messages are polynomials and the verifier has oracle access to the evaluation of polynomials on the queried points.

The following lemma is very useful in the PIOP context.

Lemma 1 (Schwartz-Zippel Lemma). *Let $P(X)$ be a non-zero polynomial of degree d over a field $\mathbb{F}$. Let S be a finite subset of $\mathbb{F}$ and let r be selected randomly from S. Then:*

$$\Pr[P(r) = 0] \leq d/|S|$$

We also require the following claim, which complements the Schwarz-Zippel lemma as a technical tool in one of our proofs.

Lemma 2. (PLONK [4], Claim 4.6) *Fix $F_1, \cdots, F_k \in F^{<n}[X]$, and $Z \in F^{<n}[X]$. Suppose that for some $i \in [k]$, $Z \nmid F_i$. Assuming Z decomposes to distinct linear factors over $\mathbb{F}$, then except with probability $k/|\mathbb{F}|$ over uniform $\alpha \in \mathbb{F}$, Z doesn't divide $G := \sum_{i=1}^{k} \alpha^{i-1} F_i$*

3.4 Roots of Unity and Lagrange Polynomials

Let $\mathbb{F}_q$ be a finite field of prime order q. We call the roots of the polynomials $Z_k(X) = X^k - 1$ the k-th roots of unity. Together, they form a multiplicative subgroup Ω_k of $\mathbb{F}_q^*$, provided that $k|(q-1)$. We say that $Z_k(X) = X^k - 1$ is the vanishing polynomial of Ω_k.

We assume that k is a power of 2, for each subgroup $\Omega_{k'}$ of Ω_k (thus, $k'|k$), we have $\omega' = \omega^{k/k'}$ where ω and ω' are the generators of Ω_k and $\Omega_{k'}$ (res.).

For any subgroup Ω_k, the collection of polynomials given by $(\mathcal{L}_{u,\Omega_k}(X))_{u \in \Omega_k}$ forms the Lagrange basis for polynomials of degree $k-1$ where,

$$\forall u \in \Omega_k : \mathcal{L}_{u,\Omega_k}(X) = \frac{u(X^k - 1)}{k(X - u)}$$

3.5 Reed-Solomon Codes

Definition 5 (Linear Code [33]). *A linear error-correcting code with message length k and codeword length n with $k < n$ is a linear subspace $C \subset \mathbb{F}^n$, such that there exists an injective mapping from message to codeword $EC : \mathbb{F}^k \to C$ which is called the encoder of the code. Any linear combination of codewords is also a codeword. The rate of the code is defined as $\rho := k/n$. The distance between two codewords u, v is the number of coordinates on which they differ, denoted as the Hamming distance $\Delta(u, v)$. The relative (or fractional Hamming distance) is defined as $\delta(u, v) = \Delta(u, v)/n$. The minimum distance is $d := \min_{u,v} \Delta(u, v)$.*

Definition 6 (Reed-Solomon Code). *Consider positive integers n, k, a finite field $\mathbb{F}$, and a set $D \subseteq \mathbb{F}^*$ with $|D| = n$ (the set D will be referred to as the domain). The Reed-Solomon code over $\mathbb{F}$ with domain D and the message space of size k is defined as:*

$$\mathsf{RS}[\mathbb{F}, D, k] := \{p(x)|_{x\in D} : p(X) \in \mathbb{F}[X], \deg(p) \leq k\},$$

By $p(x)|_{x\in D}$, we denote the set of evaluations of p over the set D and $n = |D|$ is called the codeword size. For $v \in D$ and $p \in \mathsf{RS}[\mathbb{F}, D, k]$, we will also use the notation $p|_v$ to refer to $p(v)$.

By $F^{<n}[X]$, we denote the set of polynomials of degree less than or equal to k, i.e.

$$\mathbb{F}_{<k} := \{p(X) \in \mathbb{F}[X] : \deg(p) \leq k\},$$

Distance to a Reed-Solomon Code Consider arbitrary $f \in \mathbb{F}^{|D|}$. The distance of f from the set $V = \mathsf{RS}[\mathbb{F}, D, k]$ is defined as $\Delta(f, V) := \min_{v\in V} \Delta(f, v)$ (and similarly for relative distance).

3.5.1 Reed-Solomon Codes over Roots of Unity In this work, we choose the domain set $D = \Omega_n$ as the set of n^{th} roots of unity. Consider a fixed generator ω of Ω_k. Then $D = \{\omega^i\}_{i=0}^{n-1}$ and we will associate polynomial evaluations $p(x)|_D$, called codeword space, with vectors $(p(\omega^0), p(\omega^1) \dots p(\omega^{n-1}))$, ordered by the natural ordering induced by the exponents of generator ω.

3.5.2 Interpolation By $\mathsf{Int}_c(X)$ we denote the interpolated polynomial corresponding to a Reed-Solomon codeword c, and by $\mathsf{Int}_c(x)$ the evaluation of the interpolated polynomial at point x. Using the notation introduced in Definition 6, note that for a low-degree polynomial $p(X)$, a codeword was defined as $c = p(x)_{x\in D}$. Conversely, $p(X) = \mathsf{Int}_c(X)$.

3.5.3 Quotienting For a word $f \in \mathbb{F}^n$, the quotient function $\mathsf{Quot}_{x,y} : \mathbb{F}^n \to \mathbb{F}^k$ is defined as:

$$\mathsf{Quot}_{x,y}(f_1 \dots f_n) := \Big(\mathsf{Quot}_{x,y,1}(f_1) \dots \mathsf{Quot}_{x,y,n}(f_n)\Big),$$

where $\mathsf{Quot}_{x,y,i} : \mathbb{F} \to \mathbb{F}$ and:

$$\mathsf{Quot}_{x,y,i}(f) = \frac{f_i - y}{w^i - x}$$

Note that for any $v \in \mathbb{F} \setminus D$ and any codeword c, if it holds that $c|_v = y$ (i.e. $\mathsf{Int}_c(v) = y$) then $\mathsf{Quot}_{v,y}(c)$ is well defined. Moreover, if for a codeword c we have that $\mathsf{Quot}_{v,y}(c)$ is well defined, then it must be that $\mathsf{Int}_c(v) = y$.

Lemma 3 (Quotienting Lemma). *Consider arbitrary $x \in \mathbb{F} \setminus D$ and the function $\mathsf{Quot}_{x,y} : \mathbb{F}^n \to \mathbb{F}^k$ as defined directly above.*

For any $c \in \mathbb{F}^n$ and arbitrary proximity parameter $\theta \in (0, 1)$, it holds that:

$$\delta(c, \mathsf{RS}[\mathbb{F}, D, k]) \leq \theta,\ \mathsf{Int}_c(x) = y$$
if and only if
$$\delta(\mathsf{Quot}_{x,y}(c), \mathsf{RS}[\mathbb{F}, D, k-1]) \leq \theta.$$

3.5.4 Quotienting Matrices Consider $y \in \mathbb{F}^m$. The quotienting function can be extended to any matrix $M \in \mathbb{F}^{m\times n}$ by considering every row m_i of M to be a word in $\mathbb{F}^n$. The quotient is applied on each row separately to obtain quotient matrix Q, where the (i, j) entry $Q_{i,j}$ is defined as:

$$Q_{i,j} = \mathsf{Quot}_{x,y_i,j}(M_{i,j}) = \frac{M_{i,j} - y_i}{w^j - x}$$

3.6 List Polynomial Commitments

The following theorem is widely used to justify the proximity of a batch of vectors to the codewords. It informally says that if the linear combination $f := \sum \lambda^i f_i$ has relative distance at most θ from a codeword, then each f_i follows a similar proximity property.

Theorem 1. *(Correlated Agreement Theorem (CAT) full version of [10], Theorem 6.1 and 6.2) Let the Reed-Solomon code $\mathsf{RS}[\mathbb{F}, D, k]$ have the rate ρ. Given the proximity parameter θ and the words $f_0, \ldots, f_{N-1} \in \mathbb{F}^D$ for which:*

$$\Pr_{\lambda \in \mathbb{F}} \left[\delta(\sum_i \lambda^i f_i, \mathsf{RS}) \leq \theta \right] > \epsilon$$

where θ, ϵ are from a chosen decoding regime Definition 7

Then there exist the polynomials $p_i(X) \in \mathsf{RS}[\mathbb{F}, D, k]$, and a (agreement) set $A \subset D$ of density $|A|/|D| \geq 1 - \theta$ on which $f_0, \ldots, f_{N-1}$ jointly coincide with $p_0(X), \ldots, p_{N-1}(X)$ respectively. In particular,

$$\forall \lambda \in \mathbb{F} \ : \ \delta(\sum_i \lambda^i f_i, \mathsf{RS}) \leq \theta$$

Definition 7. *(Decoding Regime [10]) Theorem 1 holds in the following decoding regimes*

- *Unique Decoding Regime: for $\theta \in (0, \frac{1-\rho}{2})$, Theorem 1 holds for $\epsilon = (N-1)|D|/|\mathbb{F}|$.*
- *List Decoding Regime: For $\theta \in (\frac{1-\rho}{2}, 1 - \sqrt{\rho})$ and setting $\theta = 1 - \sqrt{\rho} \cdot (1 + 1/2m)$ with $m \geq 3$, Theorem 1 holds for:*

$$\epsilon = (N-1) \frac{(m + 1/2)^7}{3\rho^{3/2}} \frac{|D|^2}{|F|} \tag{1}$$

  .
- *Capacity Regime: for $\theta \in (1 - \sqrt{\rho}, 1 - \rho)$, the error ϵ is* **conjectured** *to satisfy,*

$$\epsilon \leq \frac{1}{(\tau\rho)^{c_1}} \cdot \frac{(N \cdot |D|)^{c_2}}{|\mathbb{F}|}$$

  for $\tau = 1 - \rho - \theta$ and c_1, c_2 constants.

As a small remark, the last case is currently a conjecture and will not be addressed in this paper.

We say that the Reed-Solomon code $V := R[\mathbb{F}, D, k]$ is (θ, L)-list-decodable if for every $u \in \mathbb{F}^n$, there are no more than L codewords of V that are within relative Hamming distance at most θ from u.

Theorem 2. *(Johnson bound [8]) For every $\tau \in (0, 1 - \sqrt{\rho})$, the code V is $(1 - \sqrt{\rho}, 1/2\tau\sqrt{\rho})$-list-decodable.*

We are now ready to present the syntax and security of the list polynomial commitment. The definitions here follow the ones from Redshift ([22]) but extended to a batched setting. Our presentation closely follows the formalization of [11, 4]. We considered batched openings of multiple polynomials. One difference is that we only consider openings of all these polynomials at the same evaluation point.

The list polynomial commitment has a relaxed binding property, each commitment corresponding to a list of polynomials that is determined by a distance parameter. The commitment can be opened to any of the polynomials belonging to the list. Moreover, the polynomials in the list will jointly agree on the same agreement set, similarly to Theorem 1.

Definition 8 ((Batched) List Polynomial Commitment). *A list polynomial commitment scheme is a triplet $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ that is defined w.r.t. a linear code, distance parameter θ and domain D. It satisfies:*

- $\mathsf{Setup}(1^\lambda, k)$ *generates public parameters* pp *(a structured reference string) suitable to commit to polynomials of degree $< k$. Implicitly, the parameters for encoding are included in* pp.

– $\mathsf{Commit}(\mathsf{pp}, f_1(X)\dots f_n(X))$ *outputs a commitment* C *to functions* $f_1(X)\dots f_n(X) \in \mathbb{F}[X]$
– $\mathsf{OpenEval}$ *is an IOP between a prover* P_{PC} *and a verifier* V_{PC}*, where the prover is given* n *functions* $f_1(X)\dots f_n(X) \in \mathbb{F}[X]$ *and attempts to convince the verifier of the following relation:*

$$\exists A \subset D \text{ s.t } |A| \geq (1-\theta)\cdot|D| \text{ and } \exists(P_1\dots P_n) \in (\mathbb{F}^{<k}[X])^n \text{ s.t.}$$
$$\big(P_i(x) = y_i \ \wedge \ f_i(a) = P_i(a)|_{a\in A} \text{ for all } i \in [n]\big) \wedge$$
$$\wedge\ C = \mathsf{Commit}(\mathsf{pp}, f_1\dots f_n)$$

where both parties receive the following:

- *security parameter* λ*, degree bound* k *and batch size* n*, such that* $k, n = poly(\lambda)$*.*
- *The public parameters* pp*, where* $\mathsf{pp} = \mathsf{Setup}(1^\lambda, k)$*.*
- *An evaluation point* x *and alleged openings* $y = (y_1\dots y_n)$*.*
- *Alleged commitment* C *for functions* $f_1(X)\dots f_n(X)$*.*

In addition, the verifier receives oracle access to evaluations of f_i *over* D*.*

Definition 9 (Completeness of a List Polynomial Commitment Scheme). *We say that a polynomial commitment scheme has (perfect) completeness if for any security parameter* λ*, any integers* $k, n = \mathrm{poly}(\lambda)$*, any polynomials* $P_1(X)\dots P_n(X) \in \mathbb{F}_{<k}[X]$*, arbitrary evaluation point* x *and alleged opening* y*, if* $C = \mathsf{Commit}(\mathsf{pp}, P_1(X)\dots P_n(X))$ *and* $P_i(x) = y_i$ *for all* $i \in [n]$ *then an interaction of* $(P_{\mathrm{PC}}, V_{\mathrm{PC}})$ *where* P_{PC} *runs on the aforementioned parameters will result in the verifier accepting with probability one.*

Definition 10 (Knowledge Soundness in the Random Oracle Model). *There must exist a* PPT *extractor* E *such that for every* PPT *adversary* $\mathcal{A}$ *and arbitrary degree* $k = \mathrm{poly}(\lambda)$*, the probability that* $\mathcal{A}$ *wins the following game is negligible, where the probability is taken over the coins of* Setup*,* $\mathcal{A}$ *and* V_{PC}*. Moreover, the extractor has access to the random oracle queries of* $\mathcal{A}$*:*

– $\mathcal{A}$ *receives degree* k *and* $\mathsf{pp} = \mathsf{Setup}(1^\lambda, k)$*.* $\mathcal{A}$ *outputs* C*.*
– E *receives the commitment* C *and inspects the random oracle queries made by* A *in the previous step and recovers* $f_1(X)\dots f_n(X) \in [X]$*.*
– E *applies the efficient list-decoding algorithm on all* f_i *simultaneously to obtain list* L*, defined as:*

$$L = \{(P_1(X),\dots,P_n(X)) \in (\mathbb{F}^{<k}[X])^n \text{ s.t. } \exists A \subset D, \text{ s.t. } |A| \geq |D|\cdot(1-\theta) \text{ and } f_i(a) = P_i(a)|_{a\in A}\}$$

– $\mathcal{A}$ *outputs an evaluation point* x *and claimed openings* $y := (y_i)_i$*.*
– $\mathcal{A}$ *interacts with the* V_{PC} *verifier of the* $\mathsf{OpenEval}$ *algorithm. The inputs of* $\mathcal{A}$ *for this subprotocol are* C*,* x *and* y*.*
– *The extractor may check consistency and output a set* S *of witnesses, where* $S \subseteq L$*.*
– $\mathcal{A}$ *succeeds if* V_{PC} *accepts and there exists no tuple* $(P_1(X)\dots P_n(X)) \in L$ *such that* $P_i(x) = y_i$ *for all* $i \in [n]$*.*

4 UniEval Compiler: from PIOP to UniEval PIOP

Let $\mathcal{P}$ be a PIOP protocol, where for $i \in [n], j \in S_i$, the verifier queries a polynomial P_i over a point x_j.

The aim of the compiler, presented here, is to reduce the initial PIOP to a PIOP where the oracle-given polynomials are all queried at a single random point. We will call such a PIOP scheme a UniEval PIOP, and the single query is denoted "Grail query". For any evaluation $P_i(x)$ where x is not the Grail query, the verifier gets $P_i(x)$ directly from the prover.

In this model, replacing the oracle with a polynomial commitment scheme requires a proof of the evaluation for all the polynomials at the same point i.e., over the Grail query. The positive point is that batching

at the polynomial commitment level is now more straightforward as all the polynomials are queried on the same evaluation point.

Indeed, due to this compiler, the batching over different points is done at the PIOP level. At the polynomial commitment level, we only need batching over the same point.

To build our compiler, we first present a batching technique of multiple polynomials over multiple points. We then use this protocol to compile any PIOP into a UniEval PIOP.

4.1 Multiple-Point to Single-Point Reduction

We assume a set of points T and a set of n polynomials $\{i \in [n] : P_i(X)\}$, each of degree $d_i \leq d$. Each $P_i(X)$ is queried on a set of evaluation points $S_i \subset T$. Define $R_i(X)$ as the alleged evaluations of $P_i(X)$ over the set S_i, namely, $R_i(X)$ agrees with purported $P_i(X)$ over S_i (and $R_i(X)$ is of degree $|S_i|$). The aim is to present a protocol for the relation;

$$R := \{(S_i, R_i(X); P_i(X))_i \quad \forall i \quad P_i(X)|_{S_i} = R_i(X)|_{S_i}\} \tag{2}$$

Claim. The relation R holds if and only if:

$$\forall i \in [n] : (P_i(X) - R_i(X)) \prod_{x \in T \setminus S_i} (X - x) \text{ is divided by } \prod_{x \in T} (X - x). \tag{3}$$

Knowing this fact, in Fig. 1 we present our batching protocol for the relation Eq. (2). The protocol is inspired by the batching approach presented in [11].

$\mathsf{MPSP}(S_1, \cdots S_n, R_1, \cdots, R_n; P_1, \cdots, P_n)$

1. the prover sends oracle access to P_i.
2. The verifier samples $\alpha \leftarrow\$ \mathbb{F}$.
3. The prover computes and sends oracle-access to:

$$Q(X) = \sum_{i \in [n]} \alpha^i \frac{P_i(X) - R_i(X)}{\prod_{x \in S_i} (X - x)}$$

4. The verifier samples $z \leftarrow\$ \mathbb{F}$ and queries $P_1(z), \cdots P_n(z), Q(z)$.
5. Finally, the verifier checks that: relation in 3 is satisfied for $X = z$ i.e.,

$$Q(z) \prod_{x' \in T} (z - x') = \sum_{i \in [n]} \left(\alpha^i (P_i(z) - R_i(z)) \prod_{x'' \notin S_i} (z - x'') \right)$$

Fig. 1. Multi-point to single-point reduction procedure.

4.2 Security analysis

Here we prove that the protocol Fig. 1 is knowledge-sound for the relation Eq. (2). We show that if the verifier checks pass, the relation R holds with overwhelming probability. The following security proof is adjusted from [11].

Denote $Z_H(X) := \prod_H (X - x)$ for an arbitrary set $H \subset \mathbb{F}$, and we define also:

$$P(X) := \sum_{i=1}^{n} \alpha^{i-1} (P_i(X) - R_i(X)) Z_{T \setminus S_i}$$

By contradiction, assume that the relation R (from Eq. (3)) does not hold. Then there exists $i \in [n]$ and some $x \in S_i$, such that $P_i(x) \neq R_i(x)$, which means that x is not a root of $P_i(X) - R_i(X)$. This implies that we have:

$$\exists\ i \text{ s.t. } Z_{S_i} \nmid (P_i(X) - R_i(X)) \tag{4}$$

From Eq. (3), this is equivalent with:

$$Z_T(X) \nmid (P_i(X) - R_i(X)) \cdot Z_{T \setminus S_i}$$

Therefore, by Lemma 2, we know that with overwhelming probability:

$$Z_T(X) \nmid P(X)$$

On the other hand, if the verification check passes, we have using the definiton of $P(X)$ that,

$$Q(z) \cdot Z_T(z) = P(z).$$

Therefore, by the Schwartz-Zippel lemma w.r.t the variable $X = z$, we know that with overwhelming probability $Q(X) \cdot Z_T(X) = P(X)$ everywhere, which contradicts $Z_T(X) \nmid P(X)$.

4.3 Compiler: PIOP to UniEval PIOP

We are now ready to compile a PIOP to its UniEval version.

- For any PIOP, define its associated protocol PIOP$'$ as follows; we let all the queries in PIOP be sent directly to the prover, and let the prover respond to these queries (the prover replies with alleged values for the evaluations, without providing a proof at this stage, as that would be handled later in the protocol). Indeed PIOP$'$ is the same as PIOP where the prover also plays the role of the oracle by itself.
- By the end of an execution of PIOP$'$, we get the trace of the polynomial queries issued during PIOP$'$; the set of polynomials P_i, the points S_i, and the alleged evaluations of $P_i(X)$ over S_i which we denote by $R_i(X)$ (prover's responses).
- Now, we consider our multi-point to the single-point protocol in Fig. 1, for the statement (R_i, S_i) and the witness $P_i(X)$ from the trace. Call this protocol $\text{MPSP}(R_i, S_i; P_i(X))_i$.

The compiler first runs PIOP$'$, get the trace, and then runs $\text{MPSP}(R_i, S_i; P_i(X))_i$. The resulting PIOP is what we call UniEval-PIOP, denoted by UniEval-PIOP.

Knowledge-Soundness. Let $\epsilon_{\text{UniEval}}$, $\epsilon_{\text{PIOP}'}$ and ϵ_{MPSP} be, respectively, the soundness-error of protocols UniEval-PIOP, protocol PIOP$'$ and $\text{MPSP}(R_i, S_i; P_i(X))_i$. Then, we have, $\epsilon_{\text{UniEval}} \leq \epsilon_{\text{PIOP}'} + \epsilon_{\text{MPSP}}$.

5 Vortex, A (Batchable) Polynomial Commitment

Vortex is a variant of the commitment scheme proposed in Orion [33], Brakedown [19], and Ligero [3]. Vortex allows performing a batched argument of multiple committed polynomials evaluated over the same given point x. The main difference is that we discuss the security not in a standard binding model (relevant to the "unique decoding regime" in the coding literature) but rather in a more relaxed model (relevant to the "list decoding regime"). This point helps us improve the scheme's efficiency but brings some challenges for the PIOP transformation into AoK via Vortex, which we will address later. Vortex is described in Section 5.1.

For a matrix of size $m \cdot n = N$, the Vortex commitments and opening arguments have size $O(\sqrt{N})$. Moreover, the opening arguments have verification time $O(\sqrt{N})$.

5.1 Description of Vortex

In this subsection, we expand on the details of Vortex. We will first assume two integers m and k, denoting the number of rows and columns. Vortex allows committing to m vectors $w_i \in \mathbb{F}^k$ in a single commitment and opening them simultaneously for scalar products with a common public vector.

Let $\mathcal{H}$ be a hash function parameterized to be able to hash vectors of size (at least) m (For the instantiation we use MIMC [2] over SIS-hash[25]). We also use a *systematic*[2] Reed-Solomon $\mathcal{L}$ with message size k and codeword-size $n > k$. We denote its distance by d, its rate by $\rho = k/n$ and we name its encoding algorithm $\mathsf{Encode}_{\mathcal{C}}$. The Reed-Solomon encoding has $O(n \log n)$ encoding time and benefits from Maximal Distance Separability.

Vortex is a polynomial commitment scheme that efficiently opens multiple committed polynomials at the same point x. The protocol consists of four algorithms: Setup, Commit and OpenEval.

1. Setup is a transparent offline phase run by both the prover and verifier. During this phase, they perform precomputations involving sampling the parameters for the hash and the encoding scheme used as the *public parameters*.
2. The Commit algorithm: Let W, be the matrix whose i^{th} row is $w_i \in \mathbb{F}^k$. Thus, W has m rows and k columns. The prover encodes each row of W (noted by w_i) using the encoding function and obtains W' (which has n columns).[3] The prover then computes the hash of the columns. The value $H = h_1, \cdots, h_n$ forms the *commitment*.
3. The batch-opening phase or OpenEval is an interactive protocol where the prover runs the ProveOpening algorithm and the verifier runs the VerifyOpening. At the beginning of this phase, the prover holds W, W' and the verifier holds the *final commitment* as input. Both hold the statement x, y, with the restriction that $x \in \mathbb{F} \setminus D$.

   The prover's goal is to convince the verifier that $\forall i < m, \mathsf{Int}_{w_i}(x) = y_i$ if W is a batch of codewords w_i. The verifier then sends the random scalar β, and the prover responds with u claimed to be $u := \mathcal{B}^\top W$, if W is polynomial, where $\mathcal{B} = (1, \beta, \beta^2, \ldots, \beta^{m-1})$. Then, the verifier samples t columns $q_1, \cdots q_t$ $(q_i \leq n)$ uniformly at random, and the prover responds with $(s_1 \cdots s_t)$ chosen columns of W'. The verifier computes u' as the Reed-Solomon encoding of u and performs the following checks for all opened columns:

   - **Proximity Check:** the scalar-product $\mathcal{B}^\top s_i \stackrel{?}{=} u'_{q_i}$
   - the hash of s_i is correct and consistent with h_{q_i}.
   - **Evaluation Check:** the relation $\mathsf{Int}_u(x) \stackrel{?}{=} \mathcal{B}^\top \cdot y$ where $\mathsf{Int}_u(X)$ is the polynomial with the coefficients given in the vector u (see Section 3.5.2).

   The first check (the random combination over random columns), is used for checking the proximity of a batch in [3]. Fig. 2 sums up the above.

Constant Size Commitment. As a simple optimization over the commitment size, we apply a SNARK-friendly hash function (e.g., MiMC hash or Poseidon) over each h_i and then compute a Merkle tree over the results.

This is particularly useful for compiling PIOP to AoK via Vortex since the Vortex commitment phase will not be offline anymore and will be part of the proof. It is important to note, however, that the hash function used in the construction of the Merkle tree needs to be modeled as a random oracle for the scheme to retain extractability.

[2] This means the original block should be a sub-vector of the corresponding codeword. By "checksum", we refer to the part of a codeword, that is added beside the original block.

[3] Observe that, since the encoding procedure $\mathsf{encode}_{\mathcal{L}}$ is systematic, we have that all columns W are also columns of W'.

$\mathsf{Setup}(n, m, \mathcal{L}, \lambda) \to \mathsf{pp}$

1. Setup an instance of hash, Hash, corresponding to the security level λ
2. Choose t (the number of columns that should be opened later) to reach the security level λ
3. Runs pre-computations relative to $\mathsf{Encode}_{\mathcal{C}}$ (e.g., finding $D \subset \mathbb{F}_q$ and relevant parameters for the security level λ)
4. Collect all the computed parameters in pp and return it.

$\mathsf{Commit}(\mathsf{pp}, W) \to (h_1 \cdots h_n)$

1. Encode each row of W and obtain W'
2. Hash each column of W' to obtain $(h_1 \cdots h_n)$
3. Return $(h_1 \cdots h_n)$

$\mathsf{Open}(\mathsf{pp}, C, P(X)) \to 1/0$

1. run $C' = \mathsf{Commit}(\mathsf{pp}, W)$
2. If $C' = C$ output 1 else output 0.

$\mathsf{OpenEval}$ with statement $(l = (1, x \dots x^{n-1}), y)$, where $x \in \mathbb{F} \setminus D$

$\mathsf{ProveOpening}(\mathsf{pp}, W', l, y)$		$\mathsf{VerifyOpening}(\mathsf{pp}, H, l, y)$
	$\leftarrow \beta$	
$\mathcal{B} = (1, \beta, \beta^2 \dots \beta^{m-1})$ $u = \mathcal{B}^\top W$		
	$u \rightarrow$	
		$\mathbf{q} \leftarrow\$ [n]^t$
	$\leftarrow \mathbf{q}$	
	$(s_1, \cdots, s_t) \rightarrow$	
		$u' \leftarrow \mathsf{Encode}_{\mathcal{C}}(u)$ for $0 < i \leq t$:
$W' = \mathsf{Encode}_{\mathcal{C}}(W)$ Denote the columns of W' as $(s_1 \cdots s_{n'})$		
		$\langle s_i, \mathcal{B} \rangle \stackrel{?}{=} u'_{q_i}$ $\mathsf{Hash}(s_i) \stackrel{?}{=} h_{q_i}$ $\mathsf{Int}_u(x) \stackrel{?}{=} \mathcal{B} \cdot y.$

Fig. 2. The Vortex polynomial commitment.

Vortex List Polynomial Commitment for Long Polynomials

Here we show how to deal with the long polynomials.

The prover $\mathcal{P}$ can send a polynomial P whose degree is larger than the number of columns in W. The polynomial can be folded in several chunks $P(X) = P_0(X) + X^n P_1(X) + \cdots$. Each one of the chunks $P_i(X)$ is then inserted into W as an entire row.

To commit and open the polynomials $P(X)$ via Vortex, set W as above. The verifier can then recombine the $P_i(X)$ evaluations to obtain the $P(X)$ evaluation. This provides us with a way to switch between the definitions of batched polynomial commitments Definition 2 to a version that only commits to one polynomial.

6 Soundness of Vortex in the List-Decoding Regime

If the prover is honest and $P(X)$ is indeed a polynomial, then the proof of correctness is straightforward. Thus, we discuss soundness. We discuss the soundness of Vortex (as an LPC) based on the Correlated Agreement Theorem and in the list decoding regime.

We denote the rows of matrix W' by w'_i. Vortex aims to prove that there exist codewords g_i corresponding to low-degree polynomials $\mathsf{Int}_{g_i}(X)$ for every index $i = 1, \ldots, m$, such that $\delta(g_i|_D, w'_i) \leq \theta$ (over the domain D) and all the g_i jointly agree with w_i on agreement sets A of size $|A| \geq (1-\theta) \cdot |D|$. Moreover, we ask that for a given evaluation point x, we have $y_i = \mathsf{Int}_{g_i}(x)$.

Theorem 3. *Consider the Vortex protocol, as described in Fig. 2. The protocol satisfies knowledge soundness in the random oracle model, in the sense of Definition 10.*

Proof. First, we consider a hybrid game in which the responses of the random oracle never contain a collision. The random oracle is programmed so that on every new random oracle query it will sample a uniform output that has not been chosen before. The view of the adversary in this game is statistically close to the one in the original game, and we denote this statistical gap by $\epsilon_{\text{collision}}$.

With this simplification, for the purposes of this proof sketch, we now consider that the prover knows a matrix W' that is committed inside the hash (if it knew more preimages, that would imply that the mallicious prover can find collisions in the hash function).

By the definition of $\mathcal{R}_{LPC}$, the relation returns 0 if W' is far from any codeword, or no close codeword evaluates to the claimed y at point x.

Define the two following cases:

- **Case A:** Matrix W' is far from any matrix of codewords.
- **Case B:** Matrix W' is close to at least one matrix of codewords, but no such codeword matrix evaluates to y at point x.

The structure of the matrix is not directly apparent to the verifier. We will bound the probability of success for a malicious prover that has committed to matrix W' in each of the two cases, and then explain how the two probabilities relate to the overall knowledge soundness of the protocol.

6.1 Bounding on Case A

Define w as the linear combination over the rows of W' i.e., $w := \sum \beta^i W'_i$. We denote by $r_{\mathcal{A}}$ the random coins of the adversarial prover, and r_V the coins of the verifier. To bound soundness in Case A, define by E_{KS} the event that $\mathcal{A}$ wins in the knowledge-soundness game of Definition 10. Then, we have:

$$\begin{aligned}\Pr_{r_P, r_V}(E_{\text{KS}}) \leq &\Pr_{r_P, r_V}(E_{\text{KS}} \mid (\delta(w, \mathsf{RS}) > \theta)) \\ &+ \Pr_{r_P, r_V}(\delta(w, \mathsf{RS}) \leq \theta)\end{aligned}$$

From the Correlated Agreement Theorem, we know that the second term happens with probability ϵ, where θ and ϵ are from Correlated Agreement Theorem.

Regarding the first term, there are two cases that the test may pass;

- The (honest) value w would be different from the codeword u, and then the prover tries to change the value of the opened column by finding a collision on the RO responses. However, recall that we work in a hybrid game in which the random oracle is programmed in such a way as to never produce collisions.
- The prover is just lucky and the chosen columns are the one that matches the codeword. This happens with probability $(1-\theta)^t$.

Therefore,

$$\begin{aligned}\Pr_{r_P, r_V}(E_{\text{KS}}) \leq \Pr_{r_P, r_V}(E_{\text{KS}} \mid (\delta(w, \mathsf{RS}) > \theta)) + \Pr_{r_P, r_V}(\delta(w, \mathsf{RS}) \leq \theta) \leq \\ \leq (1-\theta)^t + \epsilon\end{aligned}$$

6.2 Bounding on Case B

Here we bound the soundness error conditioned on Case B. Here, matrix W' is close to at least one matrix of codewords, but no such codeword matrix evaluates to y at point x.

We define the list L as the set of matrix G' where i-th row of G' is at most θ-far from the i-th row of W'. Thus,

$$L := \{G' \ : \ \exists A \subset D, |A| \geq (1-\theta)|D| \ s.t. \ w'_i|_A = g'_i|_A\}$$

For a matrix $M = (m_i)_i$ of size $m \cdot n$, we use functions $\mathsf{Quot}_{x,y,j}$ (see Section 3.5.4) to define the quotient matrix Q w.r.t the point $x \in \mathbb{F}$ and the vector $y = (y_i)_i$ of length m as:

$$Q_{i,j} = \mathsf{Quot}_{x,y_i,j}(M_{i,j}) = \frac{M_{i,j} - y_i}{w^j - x}$$

Remark: When arguing that the quotients are well defined, we use that $x \in \mathbb{F} \setminus D$.

Theorem 4. *If the proximity test passes with probability more than $\epsilon^* := (1-\theta)^t + \epsilon$, and the evaluation check on u' passes, then there is a codeword $G \in L$ such that $g_i(x) = y_i$. Define by E_{KS} the event that $\mathcal{A}$ wins the knowledge soundness game of Definition 10, and by r_P, r_V the random coins of the prover and verifier. Consequently, in case B, we have*

$$\Pr_{r_P, r_V}(E_{\mathrm{KS}}) = 0$$

.

Proof. Recall that β is the coin used to compute the linear combination and $\mathbf{q}$ is the vector sent by the verifier in order to specify which columns to open. The probability bound of the hypothesis is equivalent to:

$$\Pr_{\beta \in \mathbb{F}}[\Pr_{\mathbf{q} \in [n]^t}[u'_{q_j} = (\mathcal{B} \cdot W')_{q_j} \text{ for all } j \in [t]] \geq (1-\theta)^t] \geq \epsilon \tag{5}$$

Now, we note that the verifier accepts only if the evaluation check is successful, therefore we have that $\mathsf{Int}_{u'}(x) = \mathcal{B}y$, which means that we can define the following quotient vector $q_u = \mathsf{Quot}_{x,\mathcal{B}\cdot y}(u')$ (the quotient is well defined due to the evaluation check and the fact that $x \in \mathbb{F} \setminus D$). Equivalently, q_u is such that:

$$\mathsf{Int}_{q_u}(X) := \frac{\mathsf{Int}_{u'}(X) - \mathcal{B}y}{X - x}$$

The vector q_u thus corresponds to the quotienting of the vector u' (and recall that u' was obtained on the verifier side as the encoding of the vector u sent by the prover during the protocol). Therefore, since $u' \in \mathsf{RS}[\mathbb{F}, D, k]$:

$$q_u := \mathsf{Quot}_{x,\mathcal{B}\cdot y}(u') \in \mathsf{RS}[\mathbb{F}, D, k-1]$$

.

Using Eq. (5), recall that in this sub-case, we know that the proximity check over W' passes with probability more than ϵ^* (the proximity check ensured that the columns of W' queried by the verifier correspond to u'). Therefore, we have that $\delta(u', \mathcal{B} \cdot W') \leq \theta$ for an $\epsilon \cdot |\mathbb{F}|$ portion of β. More precisely,

$$\Pr_{\beta \in \mathbb{F}}[\mathcal{B} \cdot W' = u' \text{ over some set } A, \quad |A| > (1-\theta)^t \cdot |D|] \geq \epsilon$$

From the two above equations, we conclude that for an $\epsilon \cdot |\mathbb{F}|$ portion of β,

$$\exists q_u \in \mathsf{RS}[\mathbb{F}, D, k-1] \text{ s.t. } \mathsf{Quot}_{x,\mathcal{B}\cdot Y}(\mathcal{B} \cdot W') = q_u, \text{ over } A \text{ where } |A| \geq (1-\theta)|D|$$

The key observation now concerns the role played by $\mathcal{B}$ in the computation of the quotient, by noting that $\mathsf{Quot}_{x,\mathcal{B}\cdot Y}(\mathcal{B} \cdot W') = \mathcal{B} \cdot \mathsf{Quot}_{x,Y}(W')$ to rewrite the probability as:

$$\Pr_{\beta \in \mathbb{F}}\Big(\exists q_u \in \mathsf{RS}[\mathbb{F}, D, k-1] \text{ s.t. } \mathcal{B} \cdot \mathsf{Quot}_{x,y}(W') = q_u \text{ over } A, \text{ where } |A| \geq (1-\theta)|D|\Big) \geq \epsilon$$

Since $\mathcal{B} \cdot \mathsf{Quot}_{x,y}(W')$ is the random linear combination over the rows of quotient matrix $\mathsf{Quot}_{x,y}(W')$, the condition of the correlated agreement theorem (CAT, Theorem 1) over $\mathsf{Quot}_{x,y}(W')$ is satisfied. Namely, the random linear combination over the rows of the quotient matrix is close to a codeword. Therefore by CAT, we have that each row q_i of $\mathsf{Quot}_{x,y}(W')$ satisfies that $\delta(q_i, \mathsf{RS}[\mathbb{F}, D, k-1]) < \theta$, and consequently by Lemma 3 each row of W' is close to a codeword $g_i \in \mathsf{RS}[\mathbb{F}, D, k-1]$ such that $\mathsf{Int}_{g_i}(x) = y_i$.

Taking into account all the terms, for the soundness-error of Vortex we have that for $m' \geq 3$.

$$\begin{aligned} \epsilon_{\text{soundness}} &\leq \epsilon_{\text{collision}} + (1-\theta)^t + \epsilon \\ \theta &= 1 - \sqrt{\rho} - \frac{\sqrt{\rho}}{2m'} \\ \epsilon &\leq (m-1)\frac{(m'+1/2)^7}{3\rho^{3/2}}\frac{|D|^2}{|\mathbb{F}|} \end{aligned} \tag{6}$$

where m, k are the number of rows and columns of W (choice of m also depends on the security level of the hash function, and the choice of k depends on the message space of Reed-Solomon code).

Knowledge extraction of Vortex We first recap the following lemma showing an efficient decoding algorithm for the Reed-Solomon code.

Theorem 5. *(Guruswami-Sudan) For the Reed-Solomon code $\mathsf{RS}[\mathbb{F}, D, k]$ with rate ρ and proximity parameter $\theta < 1 - \sqrt{\rho}$, the Guruswami-Sudan algorithm can recover from $w \in \mathbb{F}^n$ all the codewords c such that $\delta(w, c) \leq \theta$ from w in time $O(n^3)$.*

Let P be a prover that suceeds in the interaction with the Vortex verifier with non-negligible probability. The extractor uses this prover to extract the witness W', in the following manner: it runs the prover in the ROM model. By this, the extractor will get access to W'. To extract the matrices G, we use the approach of [31, Lemma 1] that uses the Guruswami-Sudan list decoding repeatedly over the rows of W' and for every row, it intersects the result with the agreement sets that have already been computed, such that the intersection is of size $\geq (1-\theta) \cdot |D|$. By this, it extracts all the codeword matrices close to W'. Then we can find the one consistent with u, x, y (e.g., by brute force, since the list size is polynomial).

7 Soundness of AoK from PIOP and LPC

In [14], the authors show that combining a knowledge-sound polynomial commitment with knowledge-sound PIOP results in a knowledge-sound argument system. This cannot be applied directly to our setting. Particularly, since we are working with polynomial commitments in the list decoding regime (LPC), the knowledge-soundness of Vortex is not defined w.r.t a standard relation for a PC scheme.

In [22], they show that Batch-FRI in the list-decoding regime (as an LPC) can be combined with PLONK-PIOP resulting in an argument system.

There is some evidence that shows that such a transformation can still be possible for special PIOP and with the cost of losing a factor $|L|$ of the soundness of PIOP [22, 8].

Here we generalize these results and show that any PIOP can be combined with an LPC to give a secure AoK.

Slightly more formal, let (P_O, V_O) be a PIOP for the relation $\mathcal{R}$ that is transformed to a AoK (P, V) via a list polynomial commitment (P_c, V_c). Then it is conjectured that the soundness-error of AoK follows from,

$$|L| \cdot \epsilon_{\text{PIOP}} + \epsilon_{\text{LPC}} \approx O(|L| \cdot k/|\mathbb{F}| + \epsilon_{\text{LPC}})$$

where k is the degree of polynomials involved in the PIOP and L is the maximum size of the list associated with the LPC. If the size of the field is big compared to $|L|$, working in the list decoding regime could provide useful tradeoffs.

In Section 7.2, we present a proof and bounds for the compilation to AoK from PIOP and list polynomial commitments.

7.1 Aggregatable LPC

An aggregatable LPC scheme is a LPC equipped with a batch opening algorithm. The batch opening algorithm can prove the correct evaluation of polynomials (at the same point) committed separately. Note the difference between batching the polynomials under the same commitment (Vortex as a batch polynomial commitment) and batching under different commitments which we call aggregation here.

Apart from the batching property, what makes aggregatable-LPC interesting is the fact that it imposes the same agreement set (out of $|L|$ sets) for all the polynomials committed separately (e.g., committed in different rounds of PIOP). This will allow us to avoid a soundness loss of $|L|^r$, where r is the number of rounds in the PIOP. Instead, the soundness loss will be $|L|$.

Aggregation of Vortex. Regarding Vortex, its Aggregation is similar to the original Vortex scheme. Let W_i denote the matrices committed separately. Generally speaking, the witness matrix W is built by putting the matrices W_i one over the other and then applying the Vortex over W. The differences and the details are described in the following.

- set the parameters for Vortex.
- commit to each matrix W_i separately by Vortex commitment and denote it as c_i.
- to open all the polynomials at point x, Set $l = (1, x, x^2, \ldots, x^n)$ and let W be the stacking of the matrices W_i over each other.
  - the verifier sends the randomness β where it receives u as the response from the prover.
  - the verifier sends the random list of columns $\mathbf{q}$.
  - the prover opens the chosen columns of W, it also sends $u = \bar{\mathcal{B}} \cdot W$ where $\bar{\mathcal{B}} = (\mathcal{B}, \beta^m\mathcal{B}, \beta^{2m}\mathcal{B}, \ldots)$ and $\mathcal{B}$ is as in the Vortex.
  - the verifier extends u to its Reed-Solomon encoding and obtains u'. It then proceeds similarly to the original Vortex protocol and performs the checks as follows:
    - for the chosen columns, it breaks them into sub-columns of W_i and checks the hash consistency separately with each c_i.
    - checks $u \cdot l = \sum_i \beta^{im}\mathcal{B} \cdot \vec{y}_i$ where $\vec{y}_i$ represent the evaluation values associated with W_i (this check is equivalent to $\mathsf{Int}_u(x) = \sum_i \beta^{im}\mathcal{B} \cdot \vec{y}_i$).
    - $\sum \bar{\mathcal{B}} \cdot s_{(i)} \stackrel{?}{=} u'_{q_i}$ for the chosen columns $\{s_{(i)}\}_{i \in \mathbf{q}}$ of W.

Note that the aggregatable version of Vortex is equivalent to an application of Vortex over the big matrix W (described above). Therefore, the transformation preserves its knowledge soundness where in the soundness formula Eq. (6) the number of rows m in the last term and in ϵ is the number of rows of W, i.e., the big matrix (note that $\epsilon_{\text{collision}}$ does not change).

We emphasize that for optimizing the proof size during the compilation of PIOP to AoK, each Vortex commitment c_i is of constant size via MiMC hashing and the Merkle-Tree structure as explained in the description of Vortex.

7.2 AoK from UniEval PIOP and Aggregatable-LPC

As mentioned in the description of Vortex, a commitment can be opened to a list of size $|L|$. In order to replace the oracle of UniEval PIOP with aggregatable-LPC, one replaces the oracle with the i^{th} element of the lists. We show that aggregatable-LPC can simulate the oracle of UniEval-PIOP.

Regarding the security of such a compiler, we prove that if UniEval-PIOP and aggregatable-LPC are knowledge-sound, then the resulting AoK is knowledge-sound. Before going to the security proof, let us clarify how we combine an LPC with an UniEval-PIOP to have a more efficient AoK.

AoK Construction

Here we assume that PIOP has several polynomials for the same round i which we embed in matrix W_i. For a batch of matrices W_i from different rounds, we have to commit to each matrix separately at the associated

round but can defer the batch opening (aggregation step) once at the end. Without loss of generality, we assume the matrices W_i are all of the same size. We also assume that UniEval-PIOP has been processed through the compiler in Fig. 1, and now its verifier sends only a single query (at point z as in Section 4.1). The query is for all the polynomials at the same point, and this allows us to benefit from batching properties of Vortex over the same point (for both cases; polynomials committed at the same round, or in different rounds).

7.3 Soundness of AoK

Theorem 6. *Let aLPC be an aggregated LPC and* Prot *be an UniEval PIOP for a relation R. Consider the interactive argument of knowledge protocol* Prot' *for the same relation R, obtained in the following way:*

- *every time a polynomial is sent to an oracle in* Prot*, the operation is replaced by sending an aLPC commitment in* Prot'*.*
- Prot *contains rounds in which the prover sends alleged evaluations of polynomials without an accompanying proof (as in Section 4). The same rounds are carried out identically in* Prot'*.*
- *The final round of* Prot *represents an opening of all polynomials at the same point z. In* Prot'*, this is replaced by an opening of all the polynomials committed using the aLPC protocol at point z.*
- *All other rounds of* Prot *are reproduced identically in* Prot'*.*
- *The verifier* $V_{\mathsf{Prot}'}$ *acts in the same way as* V_{Prot} *but it also runs* V_{aLPC} *in order to check the claimed openings at the end of the protocol.*

We require that the aLPC commitment is knowledge-sound in the random oracle model and Prot *is a knowledge-sound UniEval PIOP for R. Moreover, both aLPC and* Prot *have straight-line extractors and the list of polynomial candidates can be extracted using only the commitments and the random oracle queries. Then, it holds that* Prot' *is a knowledge-sound interactive argument of knowledge for R in the random oracle model.*

Proof. We first exhibit an extractor E_{AOK} for the relation R. E_{AOK} uses the aLPC extractor E_{aLPC} and the UniEval PIOP extractor E_{PIOP}. E_{AOK} interacts with an adversary P^* against the knowledge-soundness of Prot' and proceeds as follows:

- E_{AOK} will interact with adversary P^* and make several changes in order to construct on-the-fly an adversary P_{PIOP} against the UniEval PIOP protocol. The constructed adversary P_{PIOP} will make use of access to random oracle queries and run the extractor E_{aLPC} as a subroutine.
- E_{AOK} will run the extractor E_{PIOP} on an interaction with adversary P_{PIOP} which is built on-the-fly from the interaction with P^*. E_{PIOP} will output a witness if the interaction resulted in an accepting transcript, and E_{AOK} outputs the same witness as its result.

We show how to start from adversary P^* against the knowledge soundness of the argument of knowledge Prot', and construct an adversary P_{PIOP} against the UniEval PIOP Prot. the Adversary P_{PIOP} proceeds as follows:

- It receives a statement $\mathsf{stm}_{\mathrm{AOK}}$ from P^*, and sets it as its own statement $\mathsf{stm}_{\mathrm{PIOP}}$ by forwarding it to its challenger (i.e. $\mathsf{stm}_{\mathrm{AOK}} = \mathsf{stm}_{\mathrm{PIOP}}$).
- It then honestly generates the parameters for aLPC and sends them to P^*.
- The first time P^* commits to a polynomial, P_{PIOP} uses the aLPC extractor to obtain the list of polynomials that correspond to the commitment. It aborts if the extraction fails.
  Since P^* will be able to later open to any of the polynomials in the list, P_{PIOP} guesses an index i^* that corresponds to the i^{th} polynomial in the list.
- Guessed index i^* is kept hidden from P^*. P_{PIOP} forwards the i^{th} polynomial to the PIOP oracle.
- All subsequent rounds in which P^* commitms to more polynomials require that P_{PIOP} extracts using the aLPC extractor and forwards the polynomial at index i^*.

– Verifier queries are sent to P^*. If the responses are not consistent with guess i^*, P_{PIOP} aborts. Otherwise, it forwards the responses back to the verifier.
– Finally, as in Section 4, an oracle query on point z is received from the verifier. If $z \in D$, then since the security of our aLPC is not guaranteed in this case, it aborts.
Otherwise, P_{PIOP} forwards $z \notin D$ to P^*, which replies with an aggregated aLPC proof that certifies openings for all polynomials on z. P_{PIOP} aborts if there are inconsistencies with the guessed index i^*. Otherwise, it forwards the responses to the verifier.

The prover P^* has the freedom to open to any polynomial belonging to the decoding list. However, once P^* chooses an index, it has to open any further aggregated commitments to the same index (the index corresponds to choosing an agreement set in Theorem 1). If it did not open further indices consistently, this would be detectable by the fact that a verifying aLPC opening can only be performed with respect to the same index.

There is partial leakage on index i^* from the point of view of P^*. By the fact that an abort has not yet been initiated by P_{PIOP}, P^* infers only that index i^* is consistent with its own choices of how to respond with claimed evaluations to polynomials (without an aLPC proof, which is sent only at the end). However, these choices are also enforced to be consistent with the aLPC opening in the last round.

We define the following events:

– $\mathsf{Ev}^{\text{PIOP}}_{\text{KS},P_{\text{PIOP}}}$ is the event that P_{PIOP} wins its knowledge soundness game. This means that $\langle P_{\text{PIOP}} \leftrightarrow V_{\text{PIOP}} \rangle = 1$ for an invalid $\mathsf{stm}_{\text{PIOP}}$. Verifier V_{PIOP} plays the role of the external challenger.
– $\mathsf{Ev}^{\text{AOK}}_{\text{KS},P^*}$ is the event that $\langle P^* \leftrightarrow V_{\mathsf{Prot}'} \rangle = 1$ in the context of P^* winning the knowledge-soundness game in Definition 1.
– $\mathsf{Ev}^{\text{aLPC}}_{\text{KS},P^*}$ is the event that P^* breaks the security of the aLPC commitment, as defined in Definition 10.
– $\mathsf{Ev}^{\text{index}}_{P_{\text{PIOP}}}$ is the event that P_{PIOP} guesses the index i^* correctly.

From the manner in which we defined P_{PIOP}, and Prot', we have:

$$\Pr(\mathsf{Ev}^{\text{PIOP}}_{\text{KS},P_{\text{PIOP}}}) = \Pr(\mathsf{Ev}^{\text{AOK}}_{\text{KS},P^*} \ \wedge \ \overline{\mathsf{Ev}^{\text{aLPC}}_{\text{KS},P^*}} \ \wedge \ \mathsf{Ev}^{\text{index}}_{P_{\text{PIOP}}} \ \wedge z \notin D)$$

Since the choice of the index is independent of the success probabilities, we then have:

$$\begin{aligned}
\Pr(\mathsf{Ev}^{\text{PIOP}}_{\text{KS},P_{\text{PIOP}}}) &= 1/|L| \cdot \Pr\left(\mathsf{Ev}^{\text{AOK}}_{\text{KS},P^*} \ \wedge \ (\overline{\mathsf{Ev}^{\text{aLPC}}_{\text{KS},P^*} \vee z \in D})\right) \\
&\geq 1/|L| \cdot \left(\Pr(\mathsf{Ev}^{\text{AOK}}_{\text{KS},P^*}) - \Pr(\mathsf{Ev}^{\text{aLPC}}_{\text{KS},P^*} \vee z \in D)\right) \\
&\geq 1/|L| \cdot \left((\Pr(\mathsf{Ev}^{\text{AOK}}_{\text{KS},P^*}) - \Pr(\mathsf{Ev}^{\text{aLPC}}_{\text{KS},P^*}) - \Pr(z \in D)\right)
\end{aligned}$$

Therefore, we equivalently have:

$$\Pr(\mathsf{Ev}^{\text{AOK}}_{\text{KS},P^*}) \leq |L| \cdot \Pr(\mathsf{Ev}^{\text{PIOP}}_{\text{KS},P_{\text{PIOP}}}) + \Pr(\mathsf{Ev}^{\text{aLPC}}_{\text{KS},P^*}) + \Pr(z \in D)$$

Since all the probabilities on the right are negligible, $\Pr(\mathsf{Ev}^{\text{AOK}}_{\text{KS},P^*})$ is negligible.

Future Work

We plan to add an analysis of the transformation to non-interactive argument of knowledge, concrete parameter choices for the entire scheme and benchmarks in future versions of this document.

References

[1] *A native zkEVM Layer 2 Solution for Ethereum*. URL: https://scroll.io/.

[2] Martin R. Albrecht et al. "MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity". In: *ASIACRYPT*. Vol. 10031. LNCS. 2016, pp. 191–219.

[3] Scott Ames et al. "Ligero: Lightweight Sublinear Arguments Without a Trusted Setup". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (2017).

[4] Zachary Ariel Gabizon, J. Williamson, and Oana Ciobotaru. "PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge". In: *IACR Cryptol. ePrint Arch.* (2019), p. 953.

[5] Olivier Bégassat et al. *A ZK-EVM specification*. 2022.

[6] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. "Interactive Oracle Proofs". In: *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*. Ed. by Martin Hirt and Adam D. Smith. Vol. 9986. Lecture Notes in Computer Science. 2016, pp. 31–60. DOI: 10.1007/978-3-662-53644-5_2. URL: https://doi.org/10.1007/978-3-662-53644-5%5C_2.

[7] Eli Ben-Sasson et al. "Aurora: Transparent Succinct Arguments for R1CS". In: *IACR Cryptol. ePrint Arch.* 2018.

[8] Eli Ben-Sasson et al. "DEEP-FRI: Sampling Outside the Box Improves Soundness". In: *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*. Vol. 151. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, 5:1–5:32. DOI: 10.4230/LIPICS.ITCS.2020.5. URL: https://doi.org/10.4230/LIPIcs.ITCS.2020.5.

[9] Eli Ben-Sasson et al. "Fast Reed-Solomon Interactive Oracle Proofs of Proximity". In: *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018*. Vol. 107. LIPIcs. 2018, 14:1–14:17.

[10] Eli Ben-Sasson et al. "Proximity Gaps for Reed-Solomon Codes". In: *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*. Ed. by Sandy Irani. IEEE, 2020, pp. 900–909. DOI: 10.1109/FOCS46700.2020.00088.

[11] Dan Boneh et al. "Efficient polynomial commitment schemes for multiple points and polynomials". In: *IACR Cryptol. ePrint Arch.* (2020), p. 81.

[12] Jonathan Bootle, Alessandro Chiesa, and Jens Groth. "Linear-Time Arguments with Sublinear Verification from Tensor Codes". In: *IACR Cryptol. ePrint Arch.* 2020 (2020), p. 1426.

[13] Sean Bowe, Jack Grigg, and Daira Hopwood. *Recursive Proof Composition without a Trusted Setup*. Cryptology ePrint Archive, Report 2019/1021. 2019.

[14] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. "Transparent SNARKs from DARK Compilers". In: *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12105. Lecture Notes in Computer Science. Springer, 2020, pp. 677–706.

[15] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. "Fractal: Post-quantum and Transparent Recursive Proofs from Holography". In: *LNSC*. Vol. 12105. May 2020, pp. 769–793.

[16] Alessandro Chiesa et al. "Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS". In: *Advances in Cryptology EUROCRYPT*. Vol. 12105. LNCS. Springer, 2020, pp. 738–768.

[17] the Electric Coin Company. *The Halo 2 book*. URL: https://zcash.github.io/halo2/.

[18] Lior Goldberg, Shahar Papini, and Michael Riabzev. "Cairo - a Turing-complete STARK-friendly CPU architecture". In: *IACR Cryptol. ePrint Arch.* 2021 (2021), p. 1063.

[19] Alexander Golovnev et al. "Brakedown: Linear-time and post-quantum SNARKs for R1CS". In: *IACR Cryptol. ePrint Arch.* 2021 (2021), p. 1043.

[20] Jens Groth. "On the Size of Pairing-Based Non-interactive Arguments". In: *Advances in Cryptology - EUROCRYPT*. Vol. 9666. LNCS. Springer, 2016, pp. 305–326.

[21] Ulrich Haböck. *A summary on the FRI low degree test*. Cryptology ePrint Archive, Paper 2022/1216. https://eprint.iacr.org/2022/1216. 2022. URL: https://eprint.iacr.org/2022/1216.

[22] Assimakis Kattis, Konstantin Panarin, and Alexander Vlasov. "RedShift: Transparent SNARKs from List Polynomial Commitment IOPs". In: *IACR Cryptol. ePrint Arch.* (2019), p. 1400.

[23] Abhiram Kothapalli, Srinath T. V. Setty, and Ioanna Tzialla. "Nova: Recursive Zero-Knowledge Arguments from Folding Schemes". In: *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part IV*. Vol. 13510. Lecture Notes in Computer Science. Springer, 2022, pp. 359–388. URL: https://doi.org/10.1007/978-3-031-15985-5%5C_13.

[24] Benoît Libert, Somindu C. Ramanna, and Moti Yung. *Functional Commitment Schemes: From Polynomial Commitments to Pairing-Based Accumulators from Simple Assumptions*. 2016.

[25] Vadim Lyubashevsky et al. "SWIFFT: A Modest Proposal for FFT Hashing". In: *Fast Software Encryption, 15th International Workshop, FSE 2008*. Vol. 5086. Lecture Notes in Computer Science. Springer, 2008, pp. 54–72. DOI: 10.1007/978-3-540-71039-4_4. URL: https://doi.org/10.1007/978-3-540-71039-4%5C_4.

[26] Mary Maller et al. "Sonic: Zero-Knowledge SNARKs from Linear-Size Universal and Updatable Structured Reference Strings". In: *ACM SIGSAC -CCS*. ACM, 2019, pp. 2111–2128.

[27] I. S. Reed and G. Solomon. "Polynomial Codes Over Certain Finite Fields". In: *Journal of the Society for Industrial and Applied Mathematics* 8.2 (1960), pp. 300–304. DOI: 10.1137/0108018. eprint: https://doi.org/10.1137/0108018. URL: https://doi.org/10.1137/0108018.

[28] Srinath Setty. *Spartan: Efficient and general-purpose zkSNARKs without trusted setup*. CRYPTO. 2020.

[29] Srinath T. V. Setty, Justin Thaler, and Riad S. Wahby. "Unlocking the lookup singularity with Lasso". In: *IACR Cryptol. ePrint Arch.* (2023), p. 1216. URL: https://eprint.iacr.org/2023/1216.

[30] Polygon Hermez Team. *Scalable payments. Decentralised by design, open for everyone*. https://hermez.io.

[31] StarkWare Team. "ethSTARK Documentation". In: *IACR Cryptol. ePrint Arch.* (2021), p. 582.

[32] Tiacheng Xie et al. "Libra: Succinct Zero-Knowledge Proofs with Optimal Prover Computation". In: *Advances in Cryptology - CRYPTO*. Vol. 11694. LNCS. Springer, 2019, pp. 733–764.

[33] Tiancheng Xie, Yupeng Zhang, and Dawn Xiaodong Song. "Orion: Zero Knowledge Proof with Linear Prover Time". In: *IACR Cryptol. ePrint Arch.* 2022.

[34] Risc Zero. https://github.com/risc0/risc0. 2022.

[35] Jiaheng Zhang et al. *Transparent Polynomial Delegation and Its Applications to Zero Knowledge Proof*. 2020 IEEE Symposium on Security and Privacy (SP). 2019.