$\mathscr{f}\mathscr{f}\mathfrak{lon}\mathscr{K}$: a Fast-Fourier inspired verifier efficient version of $\mathscr{P}\mathfrak{lon}\mathscr{K}$

Ariel Gabizon Zachary J. Williamson
Aztec Network

July 26, 2022

Abstract

We present a variant of the Kate, Zaverucha and Goldberg polynomial commitment scheme [KZG10] where d polynomials can be opened at a point that is a d'th power, such that the amount of verifier group operations does not depend on d. Our method works by reducing opening multiple polynomials at a single point x, to opening a single polynomial at many points via an "FFT-like identity".

As an application we present a version of the $\mathscr{P}\mathfrak{lon}\mathscr{K}$ zk-SNARK[GWC19] with significantly improved verifier performance, at the cost of roughly tripling the prover time. Specifically, in addition to the two pairings, the verifier only performs five scalar multiplications, rather than 16 or 18 as in the versions presented in [GWC19].

1 Introduction

Polynomial commitment schemes (PCS)[KZG10] have become a central ingredient in recent constructions of succinct arguments(SNARKs) [MBKM19, Gab19, CHM+19, GWC19, BFS19] when one desires a "universal and updatable" setup procedure [GKM+]. They "force" a prover to answer verifier queries according to a fixed polynomial of bounded degree.

In blockchains such as Ethereum, the precise cost of verifiying a zk-SNARK is of crucial importance to applications such as "zk-rollups"[But]. In these recent constructions this cost mostly reduces to the verification cost of the open procedure of the PCS. In this procedure we verify the correctness of evaluations given to the verifier, of polynomials previously committed by the prover. In this work we give a novel method to reduce verifier cost when opening many commitments in a [KZG10]-style PCS.

1.1 Overiew of method and comparison to previous techniques:

The original scheme of [KZG10] requires two pairings to open a polynomial f at a point $x \in \mathbb{F}$. If we wish to open several polynomials $f_0, \ldots, f_{t-1}$ at x - using [KZG10] directly would require $2t$ verifier pairings. The common way to improve on this, used in

[MBKM19, CHM+19, GWC19], has been to choose a random $\gamma \in \mathbb{F}$, and instead only verify the value of $f(x) := \sum_{i=0}^{t-1} \gamma^i f_i(x)$.

What verifier efficiency does this result in? The verifier only does two pairings as when opening a single polynomial. However, she must create the commitment for f out of the commitments of the $\{f_i\}$, which requires $t-1$ scalar multiplications to multiply the commitments by the scalars $\{\gamma^i\}_{i\in\{0,\ldots,t-1\}}$. Can we get rid of this dependence on t in the verifier performance?

The work of Boneh, Drake, Fisch and Gabizon [BDFG20] suggests a route: They give an opening protocol for multiple points where the number of verifier group operations only depends on the number of polynomials but not the number of points (there is still a dependency in the number of verifier field operations, but these are 3 orders of magnitude cheaper than a scalar multiplication). Thus, if we could reduce opening many polynomials at a single point to *opening a single polynomial at mulitiple points* we could then use [BDFG20] to obtain our desired result.

An illustration Suppose for a moment we only have two polynomials f_0, f_1 to open at x. A straightforward attempt to avoid the scalar multiplication would be to only open $f_0 + f_1$ at x. Let $a := f_0(x)$ and $b := f_1(x)$. This would prove that the sum of values $(f_0 + f_1)(x) = c = a + b$ is correct. However, it doesn't constrain a, b individually: For any value $a' \in \mathbb{F}$ we could choose b' such that $a' + b' = c$, and the verifier would also accept (a', b'). We thus need a way to generate another linear constraint on (a, b) *without resorting to using two polynomials.*

The well-known "FFT equation" comes to our aid. In the FFT setting, we represent a polynomial f by two polynomials f_0, f_1 of half the degree derived from its even and odd powers:

$$f(X) = f_0(X^2) + X \cdot f_1(X^2).$$

Here, we use this equation in the reverse direction - starting from f_0, f_1 and deriving f. Suppose $x = z^2$ is a square. f will allow us to derive the desired second constraint on a, b. Specifically, we open f at $\{z, -z\}$. We have

$$b_0 = f(z) = f_0(x) + z f_1(x) = a + zb$$

$$b_1 = f(-z) = f_0(x) - z f_1(x) = a - zb$$

Thus, these two openings of f have given us the desired two independent constraints on a, b and we can determine them. Using the natural extension to t'th roots of unity gives us the same thing for t polynomials.

1.2 Our results:

We compare the performance of our PCS to a more straightforward batched version of the [KZG10] scheme as in [GWC19]. For simplicity, we look at the case where we want to open t polynomials of degree smaller than n at a single point $x \in \mathbb{F}$ that is a t'th power, for $t|(|\mathbb{F}| - 1)$. The table clearly shows the tradeoff - while the verifier group operations

for opening do not depend anymore on t, the prover's do - as opposed to more standard batching where the prover's group exponentations[1] only depend on the maximal degree amongst the polynomials. See Theorem 5.2 for the more detailed efficiency properties in the general case (where each polynomial is opened at an arbitrary subset of points).

Table 1: Comparison of opening t polynomials of degree smaller than n, at a point $x \in \mathbb{F}$ of the form $x = z^t$ for some $z \in \mathbb{F}$. In prover/verifier work columns $\mathbb{G}_i$ means scalar multiplication in $\mathbb{G}_i$, $\mathbb{F}$ means addition or multiplication in $\mathbb{F}$, and $\mathbf{P}$ means pairing.

	SRS size	**prover work**	**proof length**	**verifier group operations**
KZG	n $\mathbb{G}_1$, 2 $\mathbb{G}_2$	tn $\mathbb{G}_1$, $O(tn)$ $\mathbb{F}$	t $\mathbb{G}_1$	$2t$ $\mathbf{P}$
Batched KZG as in [MBKM19, CHM+19, GWC19]	n $\mathbb{G}_1$, 2 $\mathbb{G}_2$	n $\mathbb{G}_1$,$O(tn)$ $\mathbb{F}$	1 $\mathbb{G}_1$	$t-1$ $\mathbb{G}_1$, 2 $\mathbf{P}$
This work	tn $\mathbb{G}_1$, 2 $\mathbb{G}_2$	$2tn$ $\mathbb{G}_1$, $O(tn)$ $\mathbb{F}$	2 $\mathbb{G}_1$	3 $\mathbb{G}_1$, 2 $\mathbf{P}$

Application to $\mathscr{P}\mathfrak{lon}\mathscr{K}$: The $\mathscr{P}\mathfrak{lon}\mathscr{K}$ proving system [GWC19] allows generating proofs of knowledge for assignments to fan-in two arithmetic circuits with a universal and updatable SRS (see the paragraph on this topic in Section 2.1). Plugging in our PCS into $\mathscr{P}\mathfrak{lon}\mathscr{K}$ allows saving in verifier work at the expense of increased prover computation. We compare the $\mathscr{P}\mathfrak{lon}\mathscr{K}$ scheme when using the [KZG10]-based PCS in [GWC19] and the PCS of this paper in Table 2.

Table 2: Comparison of $\mathscr{P}\mathfrak{lon}\mathscr{K}$ efficiency for fan-in two circuit with n gates.

	SRS size	**prover group operations**	**proof length**	**verifier group operations**
[GWC19]	$3n$ $\mathbb{G}_1$, 2 $\mathbb{G}_2$	$11n$ $\mathbb{G}_1$	7 $\mathbb{G}_1$, 7 $\mathbb{F}$	16 $\mathbb{G}_1$, 2 $\mathbf{P}$
this work	$9n$ $\mathbb{G}_1$, 2 $\mathbb{G}_2$	$35n$ $\mathbb{G}_1$	4 $\mathbb{G}_1$, 15 $\mathbb{F}$	5 $\mathbb{G}_1$, 2 $\mathbf{P}$

When is it worth it?

The zk-rollup setting motivates verifier-prover tradeoffs such as in this paper. We typically have "client proofs" computed by weak machines. These proofs are not posted on the blockchain, but usually only recursively verified by another SNARK. Thus, for these it makes sense to optimize prover efficiency at the expense of the verifier. On the other hand, the final proof put on chain is typically computed by a powerful machine, and is expensive to verify - since all network nodes must do so. For such proofs, it could be a good tradeoff to use the scheme of this paper.

[1] Following (perhaps faulty) conventions, we interchangeably use the notions group exponetiation and scalar multiplication.

2 Preliminaries

2.1 Terminology and conventions

We assume our field $\mathbb{F}$ is of prime order. We denote by $\mathbb{F}_{<d}[X]$ the set of univariate polynomials over $\mathbb{F}$ of degree smaller than d. In expressions involving both polynomials and constants, we will write $f(X)$ instead of f for to distinguish the two; but in contexts where it is clear f is a polynomial, we will simply write f for brevity.

We assume all algorithms described receive as an implicit parameter the security parameter λ.

Whenever we use the term "efficient", we mean an algorithm running in time $\mathsf{poly}(\lambda)$. Furthermore, we assume an "object generator" $\mathcal{O}$ that is run with input λ before all protocols, and returns all fields and groups used. Specifically, in our protocol $\mathcal{O}(\lambda) = (\mathbb{F}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, e, g_1, g_2, g_t)$ where

- $\mathbb{F}$ is a prime field of super-polynomial size $r = \lambda^{\omega(1)}$.
- $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t$ are all groups of size r, and e is an efficiently computable non-degenerate pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$.
- g_1, g_2 are uniformly chosen generators such that $e(g_1, g_2) = g_t$.

We usually let the λ parameter be implicit, i.e. write $\mathbb{F}$ instead of $\mathbb{F}(\lambda)$. We write $\mathbb{G}_1$ and $\mathbb{G}_2$ additively. We use the notations $[x]_1 := x \cdot g_1$ and $[x]_2 := x \cdot g_2$.

We often denote by $[n]$ the integers $\{1, \ldots, n\}$. We use the acronym e.w.p for "except with probability"; i.e. e.w.p γ means with probability *at least* $1 - \gamma$.

Universal SRS-based public-coin protocols We describe public-coin (meaning the verifier messages are uniformly chosen) interactive protocols between a prover and verifier; when deriving results for non-interactive protocols, we implicitly assume we can get a proof length equal to the total communication of the prover, using the Fiat-Shamir transform/a random oracle. Using this reduction between interactive and non-interactive protocols, we can refer to the "proof length" of an interactive protocol.

We allow our protocols to have access to a structured reference string (SRS) that can be derived in deterministic $\mathsf{poly}(\lambda)$-time from an "SRS of monomials" of the form $\left\{[x^i]_1\right\}_{a \le i \le b}$, $\left\{[x^i]_2\right\}_{c \le i \le d}$, for uniform $x \in \mathbb{F}$, and some integers a, b, c, d with absolute value bounded by $\mathsf{poly}(\lambda)$. It then follows from Bowe et al. [BGM17] that the required SRS can be derived in a universal and updatable setup[GKM$^+$] requiring only one honest participant; in the sense that an adversary controlling all but one of the participants in the setup does not gain more than a $\mathsf{negl}(\lambda)$ advantage in its probability of producing a proof of any statement.

For notational simplicity, we sometimes use the SRS srs as an implicit parameter in protocols, and do not explicitly write it.

3 Notation, definitions and operations on vectors and polynomials

Formally describing this paper's scheme in general form requires addressing opening multiple commitments, each committing to multiple polynomials, each opened at multiple points. To avoid this leading to very cumbersome notation and a "nightmare of indices", we define some operations on vectors and polynomials that enable more concise writing.

For positive integer t we denote by $[_{<}t]$ the integers $\{0,\ldots,t-1\}$. We use the convention for running indicies that the notation $i<t$ e.g. in $\sum_{i<t}$ means the sum is over $i\in[_{<}t]$.

Vector notation: Let D be some domain. We denote the set of vectors over D by $D^{(1)}$, and similarly, the set of vectors of vectors, and vectors of vectors of vectors by $D^{(2)}=(D^{(1)})^{(1)}$ and $D^{(3)}=(D^{(2)})^{(1)}$ respectively.

As suggestive notation, we denote elements of these sets by a corresponding number of overhead bars respectively; e.g. $\bar{S}\in\mathbb{F}^{(1)},\bar{\bar{S}}\in\mathbb{F}^{(2)}$ and $\bar{\bar{\bar{S}}}\in\mathbb{F}^{(3)}$.

For a vector $\bar{f}\in D^t$, we refer to the elements of $\bar{f}$ by $f_i, 0\leq i<t$. Similarly for $\bar{\bar{f}}\in D^{(2)}$, we refer to the elements of $\bar{\bar{f}}$, which are vectors over D, by $\bar{f}_i, 0\leq i<|\bar{\bar{f}}|$; and the elements of the $\left\{\bar{f}_i\right\}$ by $\{f_{i,j}\}_{i<|\bar{\bar{f}}|,j<|\bar{f}_i|}$ $\mathbb{F}[X]:=\mathbb{F}[X]$. $\mathbb{F}_{<d}[X]$ - elements of $\mathbb{F}[X]$ of degree smaller than d.

Operations on polynomials: For $\bar{f}\in\mathbb{F}[X]^t$ and $x\in\mathbb{F}$ we denote by $\bar{f}(x)$ the vector in $\mathbb{F}^t$, $\bar{f}(x):=(f_0(x),\ldots,f_{t-1}(x))$.

For $\bar{f}\in\mathbb{F}[X]^t$ and $\bar{Z}\in\mathbb{F}^\ell$ we denote by $\bar{f}(\bar{Z})\in\mathbb{F}^{(2)}$ the two dimensional array in $(\mathbb{F}^t)^\ell$, $\bar{f}(\bar{Z}):=(\bar{f}_i(Z_j))_{j\in[_{<}\ell]}$.

For $\bar{\bar{f}}\in\mathbb{F}[X]^{(2)}$ and $\bar{\bar{Z}}\in\mathbb{F}^{(2)}$ with $|\bar{\bar{f}}|=|\bar{\bar{Z}}|$, we denote by $\bar{\bar{f}}(\bar{\bar{Z}})\in\mathbb{F}^{(3)}$ the element $\bar{\bar{f}}(\bar{\bar{Z}}):=(\bar{f}_i(\bar{Z}_i))_{i<|\bar{\bar{f}}|}$.

4 Polynomial commitment schemes

We define polynomial commitment schemes similarly to [GWC19, BDFG20]. However, we make two modifications that enable capturing the scheme of this paper

- We allow the commit phase to take a *vector* of polynomials as input rather than just one. Although less general, for notational simplicity we allow the set of opening points to depend only on the vector in which the polynomial was committed in.
- We allow the scheme to be parameterized by a subset $\mathbf{S}\subset\mathbb{F}$ such that the opening procedure is only required to succeed on points from $\mathbf{S}$.

Definition 4.1. *Fix a finite subset of positive integers T and subset $\mathbf{S}\subset\mathbb{F}$. A $(T,\mathbf{S})$-polynomial commitment scheme is a 3-tuple $\mathscr{S}=(\mathsf{gen},\mathsf{com},\mathsf{open})$ such that*

- $\mathsf{gen}(d)$ *- is a randomized algorithm that given positive integer d outputs a structured reference string (SRS)* srs.

- $\mathsf{com}(t, \bar{f}, \mathsf{srs})$ - *is an algorithm that given* $t \in T$*, a vector of polynomials* $\bar{f} \in (\mathbb{F}_{<d}[X])^t$ *and an output* srs *of* $\mathsf{gen}(d)$*, returns a commitment* cm *to* $\bar{f}$*.*

- open *is a public coin protocol between parties* P *and* V*.* P *is given* $\bar{\bar{f}} \in (\mathbb{F}_{<d}[X])^{(2)}$*.* P *and* V *are both given*

  1. *Positive integer* d *and* $\mathsf{srs} = \mathsf{gen}(d)$*,*
  2. *Positive integer* r *and* $\overline{\mathsf{cm}} \in \mathbb{G}_1^r$ *- the alleged commitments to the* $\{\bar{f}_i\}$*,*
  3. *Vector* $\bar{t} \in T^r$ *- the alleged lengths of the* $\{\bar{f}_i\}$*.*
  4. $\bar{\bar{Z}} \in \mathbf{S}^{(2)}$*.*
  5. $\bar{\bar{\bar{S}}} \in \mathbb{F}^{(3)}$ *- the alleged values* $\bar{\bar{f}}(\bar{\bar{Z}})$*.*

  At the end of the protocol V *outputs* acc *or* rej*; such that*

  - ***Completeness:*** *Fix any* $\bar{t}, \bar{\bar{f}}$ *with* $\bar{f}_i \in (\mathbb{F}_{<d}[X])^{t_i}$,$\bar{\bar{Z}} \in \mathbb{F}^{(2)}, \bar{\bar{\bar{S}}} \in \mathbb{F}^{(3)}$ *such that*

    $$\bar{\bar{f}}(\bar{\bar{Z}}) = \bar{\bar{\bar{S}}}.$$

    Then if P *and* V *follow the protocol with these inputs,* V *outputs* acc *with probability* $1 - \mathsf{negl}(\lambda)$*.*
  - ***Knowledge soundness in the algebraic group model:*** *There exists an efficient* E *such that for any algebraic adversary* $\mathscr{A}$ *and any choice of* $d = \mathsf{poly}(\lambda)$ *the probability of* $\mathscr{A}$ *winning the following game is* $\mathsf{negl}(\lambda)$ *over the randomness of* $\mathscr{A}$*,* V *and* gen*.*
    1. *Given* d *and* $\mathsf{srs} = \mathsf{gen}(d)$*,* $\mathscr{A}$ *outputs* $\bar{t} \in T^r, \overline{\mathsf{cm}} \in \mathbb{G}_1^r$*.*
    2. E*, given access to the messages of* $\mathscr{A}$ *during the previous step, outputs* $\bar{\bar{f}}$ *with* $\bar{f}_i \in \mathbb{F}_{<d}[X]^{t_i}$*.*
    3. $\mathscr{A}$ *outputs* $\bar{\bar{Z}} \in \mathbf{S}^{(2)}, \bar{\bar{\bar{S}}} \in \mathbb{F}^{(3)}$*.*
    4. $\mathscr{A}$ *takes the part of* P *in the protocol* open *with the inputs* $\overline{\mathsf{cm}}, \bar{\bar{Z}}, \bar{\bar{\bar{S}}}$*.*
    5. $\mathscr{A}$ *wins if*
       - V *outputs* acc *at the end of the protocol.*
       - $f(\bar{\bar{Z}}) \neq \bar{\bar{\bar{S}}}$*.*

Notation: We usually omit d, srs *and* r*, and write* $\mathsf{open}(\bar{t}, \overline{\mathsf{cm}}, \bar{\bar{Z}}, \bar{\bar{\bar{S}}}; \bar{\bar{f}})$*.*

4.1 $\mathfrak{shplonk}$

From [BDFG20] we cite the following commitment scheme (that for historical reasons has become known as $\mathfrak{shplonk}$). The commitment procedure is identical to [KZG10]. Its crucial advantage is that the verifier group operations do not grow with the number of evaluation points .

Lemma 4.2. *There is a* $\{1, \mathbb{F}\}$*-PCS* $\mathscr{S}_{\mathfrak{shplonk}} = (\mathsf{gen}_{\mathfrak{shplonk}}, \mathsf{com}_{\mathfrak{shplonk}}, \mathsf{open}_{\mathfrak{shplonk}})$ *such that*

1. $\mathsf{gen}_{\mathfrak{shplonk}}(d)$ *is of the form: Choose uniform* $x \in \mathbb{F}$*. Output* $\mathsf{srs} = ([1]_1, [x]_1, \ldots, [x^{d-1}]_1, [1]_2, [x]_2)$.
2. *For integer* $n \leq d$ *and* $f \in \mathbb{F}_{<n}[X]$*, computing* $\mathsf{com}_{\mathfrak{shplonk}}(1, f, \mathsf{srs})$ *requires* n $\mathbb{G}_1$*-scalar multiplications.*
3. *Fix* $\bar{f}, \bar{\bar{Z}}, \bar{\bar{S}}$*. Let* $n := \max_i [\deg(f_i)]$*. Let* $k := |\bar{f}|$*. Then* $\mathsf{open}_{\mathfrak{shplonk}}\left(1^k, \overline{\mathsf{cm}}, \bar{\bar{Z}}, \bar{\bar{S}}; \bar{f}\right)$ *requires*
   (a) *2* $\mathbb{G}_1$ *elements sent from* P *to* V*.*
   (b) *at most* $2n + 1$ $\mathbb{G}_1$*-scalar multiplications of* P*.*
   (c) $k + 2$ $\mathbb{G}_1$*-scalar multiplications and 2 pairings of* V*.*

Proof. This is Lemma 4.1 from [BDFG20] - except that there V does $k + 3$ $\mathbb{G}_1$-scalar multiplications. However, it turns out the open protocol from there can be "normalized" to save one of the scalar multiplications, by dividing in many places by the constant $Z_{T\setminus S_1}(z)$ that appears as one of the scalar multipliers. We present the [BDFG20] open protocol here with this minor modification. To make the comparision with [BDFG20] easier, we momentarily use the notation from that paper - where T denotes the union of opening points for all polynomials, S_i denotes the opening set for the i'th polynomial , i ranges from one rather than zero; and r_i denotes the polynomial of degree $|S_i| - 1$ that coincides with f_i on S_i. (See [BDFG20] for more context and details.)

$\underline{\mathsf{open}_{\mathfrak{shplonk}}(\{\mathsf{cm}_i\}_{i\in[k]}, \{S_i\}_{i\in[k]}, \{r_i\}_{i\in[k]}; \{f_i\}_{i\in[k]}):}$

1. V sends a random challenge $\gamma \in \mathbb{F}$.
2. P sends $W := [(f/Z_T)(x)]_1$ where
$$f := \sum_{i\in[k]} \gamma^{i-1} \cdot Z_{T\setminus S_i}(f_i - r_i).$$
3. V sends a random evaluation point $z \in \mathbb{F}$
4. P sends $W' := \left[\frac{L(x)}{Z_{T\setminus S_1}(z)(x-z)}\right]_1$ where
$$L := \sum_{i\in[k]} \gamma^{i-1} Z_{T\setminus S_i}(z) \cdot (f_i - r_i(z)) - Z_T(z) \cdot (f/Z_T).$$
5. V outputs acc iff $e(F + zW', [1]_2) = e(W', [x]_2)$, where
$$F := \sum_{i\in[k]} \gamma^{i-1} \frac{Z_{T\setminus S_i}}{Z_{T\setminus S_1}}(z) \cdot \mathsf{cm}_i - \left[\sum_{i\in[k]} \gamma^{i-1} \frac{Z_{T\setminus S_i}}{Z_{T\setminus S_1}}(z) r_i(z)\right]_1 - \frac{Z_T}{Z_{T\setminus S_1}}(z) W.$$

Observe that in this form the coefficient of cm_1 in the last equation is one - this is what saves a verifier scalar multiplication compared to [BDFG20]. At the same time, it is straightforward to carry over the knowledge soundness proof from [BDFG20], as terms have simply changed by the constant $Z_{T\setminus S_1}(z)$. □

Remark 4.3. *[BDFG] generalize the above result regarding opening efficiency from [KZG10] to any PCS with a linearly homomorphic commitment scheme. Combining their generalization with the reduction of the next section could improve verifier efficiency in the* open *procedures of such schemes.*

5 The new commitment scheme

We define a few final operations and notations needed for presenting the scheme.

5.1 FFT-like operations on vectors and polynomials

For a vector $v \in \mathbb{F}^t$ and a point $x \in \mathbb{F}$, we denote $v(x) := \sum_{i<t} v_i x^i$.

For vectors $v, S \in \mathbb{F}^{(1)}$ we denote $v(S) := (v(x))_{x\in S}$

We define operators $\mathsf{combine}()$ and $\mathsf{decompose}()$ to group together and decompose polynomials "FFT style":

- $\mathsf{combine}_t(\bar{f}) : \mathbb{F}[X]^t \to \mathbb{F}[X]$ - given $\bar{f} \in \mathbb{F}[X]^t$ return
$$g(X) := \sum_{i<t} f_i(X^t) \cdot X^i$$
note that when $\bar{f} \in \mathbb{F}_{<d}[X]^t$ we have $\mathsf{combine}_t(\bar{f}) \in \mathbb{F}_{<d\cdot t}[X]$.
- $\mathsf{decompose}_t(g) : \mathbb{F}[X] \to \mathbb{F}[X]^t$ - given $g \in \mathbb{F}[X]$ return the unique $\bar{f} \in \mathbb{F}[X]^t$ such that
$$g(X) := \sum_{i<t} f_i(X^t) \cdot X^i$$

Note that these are injective and inverse operations. That is, for any $\bar{f} \in \mathbb{F}[X]^t$, $\mathsf{decompose}_t(\mathsf{combine}_t(\bar{f})) = \bar{f}$.

Notation regarding roots: We denote $p := |\mathbb{F}|$. For positive integer $t|(p-1)$, let $\omega_t \in \mathbb{F}$ be a fixed primitive t'th root of unity, i.e. $\omega_t^t = 1$ and $\omega_t^i \neq 1$ for $i < t$. For a t'th power $x \in \mathbb{F}$, fix $z \in \mathbb{F}$ such that $z^t = x$ and $z^i \neq x$ for $i < t$ in a standard way; e.g. take such z that has the smallest integer representative in $[<p-1]$. Now, define the vector $\mathsf{roots}_t(x) := (z\omega_t^i)_{i<t}$

The following simple lemma is the basis of our scheme.

Lemma 5.1. *Fix any $x \in \mathbb{F}, \bar{S} \in \mathbb{F}^t$ and $\bar{f} \in \mathbb{F}[X]^t$. Define $\bar{Z} := \mathsf{roots}_t(x), g := \mathsf{combine}_t(\bar{f})$ and $\bar{S}' := \bar{S}(\bar{Z})$.*

Then $\bar{f}(x) = \bar{S}$ if and only if $g(\bar{Z}) = \bar{S}'$

Proof. For $z \in \bar{Z}$, we have

$$g(z) = \sum_{i<t} f_i(z^t) z^i = \sum_{i<t} f_i(x) z^i = \bar{f}(x)(z).$$

So $g(\bar{Z}) = \bar{f}(x)(\bar{Z})$. Since distinct polynomials of degree less than t cannot agree on t points, we have that $\bar{f}(x) = \bar{S}$ if and only if $g(\bar{Z}) = \bar{S}'$. □

5.2 The new scheme

Choose a positive constant A dividing $p-1$. Let $T := \{0 < t \leq A |\ t|A\}$. Let $\mathbf{S}$ be the set of A'th powers in $\mathbb{F}$. We present the following $(T, \mathbf{S})$-polynomial commitment scheme.

1. $\mathsf{gen}(d)$ - choose uniform $x \in \mathbb{F}$. Output $\mathsf{srs} = \left([1]_1, [x]_1, \ldots, \left[x^{A\cdot(d-1)}\right]_1, [1]_2, [x]_2\right)$.
2. $\mathsf{com}(t, \bar{f}, \mathsf{srs})$ - for $t \in T$ and $\bar{f} \in (\mathbb{F}_{<d}[X])^t$. Let $g := \mathsf{combine}_t(\bar{f})$. Output $\mathsf{com}(t, \bar{f}, \mathsf{srs}) := [g(x)]_1$.
3. open: We first describe the open protocol for the simplest case of one commitment and one evaluation point.

   $\mathsf{open}\left(t, \mathsf{cm}, x, \bar{S}; \bar{f}\right)$:

   (a) P computes $g := \mathsf{combine}_t(\bar{f})$. (In practice P has this computed already from the commitment phase.)
   (b) P and V compute $\bar{Z}' := \mathsf{roots}_t(x)$ and $\bar{S}' := \bar{S}(\bar{Z}')$.
   (c) P and V engage in $\mathsf{open}_{\mathfrak{shplonk}}(1, \mathsf{cm}, \bar{Z}', \bar{S}'; g)$ and V outputs acc if and only if she does so in the execution of $\mathsf{open}_{\mathfrak{shplonk}}$.

   The general case is basically applying the same logic to each evaluation point and commitment.
   $\mathsf{open}\left(\bar{t}, \overline{\mathsf{cm}}, \bar{\bar{Z}}, \bar{\bar{S}}; \bar{\bar{f}}\right)$:

   (a) For $i < r$, P computes $g_i := \mathsf{combine}_{t_i}(\bar{f}_i)$. Let $\bar{g} := (g_i)_{i<r}$
   (b) P and V compute $\bar{\bar{Z}}'$ where $\bar{Z}'_i = \bigcup_{x \in \bar{Z}_i} \mathsf{roots}_{t_i}(x)$, and $\bar{\bar{S}}'$ where $\bar{S}'_i := \bigcup_{j<|\bar{\bar{S}}_i|} \bar{S}_{i,j}(\mathsf{roots}_{t_i}(Z_{i,j}))$.
   (c) P and V engage in $\mathsf{open}_{\mathfrak{shplonk}}(1^r, \overline{\mathsf{cm}}, \bar{\bar{Z}}', \bar{\bar{S}}'; \bar{g})$ and V outputs acc if and only if she does so in the execution of $\mathsf{open}_{\mathfrak{shplonk}}$.

Knowledge soundness: We look first at the simple case of one commitment and evaluation point. In this case $\mathscr{A}$ outputs an integer t, a $\mathbb{G}_1$-element cm, and coefficients $\{a_i\}$ such that $\mathsf{cm} = \left[\sum_{i<A(d-1)} a_i x^i\right]_1$. Let $g := \sum_{i<A(d-1)} a_i X^i$. We define the extractor E to output $\bar{f} = \mathsf{decompose}_t(g)$. An important point is that the extractor $E_{\mathfrak{shplonk}}$ used in [BDFG20] for the knowledge soundness game of $\mathsf{open}_{\mathfrak{shplonk}}$ outputs g when given $1, \mathsf{cm}, \{a_i\}$ by an adversary. We must show that $\mathscr{A}$ wins the knowledge soundness game with probability $\mathsf{negl}(\lambda)$. We will reduce to the knowledge soundness of $\mathsf{open}_{\mathfrak{shplonk}}$: We construct an adversary $\mathscr{A}'$ for $\mathsf{open}_{\mathfrak{shplonk}}$ that works as follows.

1. $\mathscr{A}'$ starts running the adversary $\mathscr{A}$ for the knowledge soundness game of open. She simulates the roles of V and E according to their correct behavior.

2. When $\mathscr{A}$ outputs t, a $\mathbb{G}_1$ element cm, and coefficients $\{a_i\}$ such that $\mathsf{cm} = \left[\sum_{i<A(d-1)} a_i x^i\right]_1$. $\mathscr{A}'$ outputs 1 and the same element cm and coefficients $\{a_i\}$.

3. Note that the extractor $E_{\mathfrak{shplonk}}$ from Lemma 4.2 for the knowledge soundness game of $\mathsf{open}_{\mathfrak{shplonk}}$ would output g at this point; and that the extractor E we defined for the knowledge soundness game of open outputs $\bar{f} = \mathsf{decompose}_t(g)$ at this point of the game with $\mathscr{A}$.

4. If $\mathscr{A}$ now outputs $x, \bar{S}$, $\mathscr{A}'$ outputs $\bar{Z}', \bar{S}'$ where $\bar{Z}' := \mathsf{roots}_t(x)$ and $\bar{S}' := S(Z')$.

5. Now we must define how $\mathscr{A}'$ behaves in $\mathsf{open}_{\mathfrak{shplonk}}(1, \mathsf{cm}, \bar{Z}', \bar{S}')$. She will behave exactly as $\mathscr{A}$ does in the call to $\mathsf{open}_{\mathfrak{shplonk}}$ which is part of the open procedure - note that this is well defined as at this point V will use the same inputs $1, \mathsf{cm}, \bar{Z}', \bar{S}'$ for the $\mathsf{open}_{\mathfrak{shplonk}}$ subprocedure.

We claim that the success probability of $\mathscr{A}'$ and $\mathscr{A}$ to win their respective knowledge soundness games is the same: By definition of the knowledge soundness game of $\mathsf{open}_{\mathfrak{shplonk}}$, $\mathscr{A}'$ wins if and only if

1. $V_{\mathfrak{shplonk}}$ outputs acc

2. $g(\bar{S}') \neq \bar{Z}'$.

By definition of V and Lemma 5.1 this is equivalent to

1. V outputs acc

2. $\bar{f}(x) \neq \bar{Z}$, for the output $\bar{f}$ of E.

Hence knowledge soundness follows from the knowledge soundness of $\mathsf{open}_{\mathfrak{shplonk}}$.

The general case of multiple commitments and evaluation points is totally analogus. In summary, we get

Theorem 5.2. *Fix positive integer A dividing $p-1$. Let T be the set of divisors of A; i.e. $T := \{t | t \leq A, t | A\}$. Let $\mathbf{S}$ be the set of A'th powers in $\mathbb{F}$, i.e. $\mathbf{S} := \left\{x \in \mathbb{F} | \exists z \in \mathbb{F} \text{ s.t. } z^t = x\right\}$.*

Then there is a $(T, \mathbf{S})$-PCS $\mathscr{S} = (\mathsf{gen}, \mathsf{com}, \mathsf{open})$ over $\mathbb{F}$ such that

1. *$\mathsf{gen}(d)$ is of the form: Choose uniform $x \in \mathbb{F}$. Output $\mathsf{srs} = \left([1]_1, [x]_1, \ldots, \left[x^{A \cdot d}\right]_1, [1]_2, [x]_2\right)$.*

2. *Let $\bar{f} \in (\mathbb{F}_{<d}[X])^t$ for $t \in T$. Suppose $n = \max_{i \in [<t]}[t \cdot \deg(f_i) + i]$. Then computing $\mathsf{com}(1, f, \mathsf{srs})$ requires $n+1$ $\mathbb{G}_1$-exponentiations.*

3. *Fix $\bar{t}, \bar{\bar{f}}, \bar{\bar{Z}}, \bar{\bar{\bar{S}}}$. Suppose $n_i = \max_{j \in [<t_i]}[t \cdot \deg(f_{i,j}) + i]$. Let $n := \max_i [n_i]$. Let $k := \sum_{i<r} t_i$. Then $\mathsf{open}\left(\bar{t}, \overline{\mathsf{cm}}, \bar{\bar{Z}}, \bar{\bar{\bar{S}}}; \bar{\bar{f}}\right)$ requires*

(a) 2 $\mathbb{G}_1$ elements sent from P *to* V.

(b) at most $2n + 1$ $\mathbb{G}_1$-exponentiations of P.

(c) $k + 3$ $\mathbb{G}_1$-exponentiations and 2 pairings of V.

6 Polynomial Protocols

At this point we move on to apply the new commitment scheme to the $\mathscr{Plon}\mathscr{K}$ proving system. We need to slightly alter some components from [GWC19] for this purpose. We warn that the following sections are hard to follow without an understanding of [GWC19].

We begin by modifying the definition of polynomial protocols from [GWC19]. The changes are:

- We enable compiling protocols using a PCS where the opening set is limited by requiring the $\{v_{i,j}\}$ below do not map out of the set.
- We explicitly track the number of rounds of the protocol, as with the PCS of this paper this ends up being a crucial parameter for verifier efficiency.
- We don't assume the verifier only checks *one* polynomial identity, and this makes the notation a little more cumbersome. The reason we could assume this from a certain point in [GWC19] is that the compilation from ranged protocols always produced one identity. But using that compilation will hurt verifier efficiency here.

Definition 6.1. *Fix positive integers d, D, r, a subset T of positive integers, a vector $\bar{t} \in T^r$, and a subset $\mathbf{S} \subset \mathbb{F}$. A* $(d, D, r, T, \bar{t}, \mathbf{S})$-polynomial protocol *over $\mathbb{F}$ is an r-round protocol between a prover $\mathrm{P}_{\mathsf{poly}}$, a verifier $\mathrm{V}_{\mathsf{poly}}$ and ideal party $\mathscr{I}$ that proceeds as follows.*

1. *The protocol definition includes a set of* preprocessed polynomials $\bar{f}_0 = (f_{0,1}, \ldots, f_{0,t_0}) \in (\mathbb{F}_{<d}[X])^{t_0}$.
2. *Each of the r rounds of interaction has the following form:*
   - *At round i, $\mathrm{P}_{\mathsf{poly}}$ sends to $\mathscr{I}$ a message $\bar{f}_i \in (\mathbb{F}_{<d}[X])^{t_i}$. If $\mathrm{P}_{\mathsf{poly}}$ sends a message not of this form, the protocol is aborted.*
   - *The verifier responds with public random coins.*
3. *Let $\bar{\bar{f}} := (\bar{f}_i)_{i<r}$ consist of the set of preprocessed polynomials together with the polynomials sent by $\mathrm{P}_{\mathsf{poly}}$. At the end of the protocol, $\mathrm{V}_{\mathsf{poly}}$ may ask $\mathscr{I}$ if certain polynomial identities holds between the polynomials in $\bar{\bar{f}}$. More specifically, each identity is of the form*

$$F_i(X) := G_i(X, h_{i,1}(v_{i,1}(X)), \ldots, h_{i,M}(v_{i,M}(X))) \equiv 0,$$

where

(a) *the* $h_{i,j}$ *are elements of* $\{f_{i,j}\}$

(b) *The* $\{v_{i,j}\}$ *are polynomials with the property that whenever* $x \in \mathbf{S}$*, we also have* $v_{i,j}(x) \in \mathbf{S}$*.*

(c) $F_i \in \mathbb{F}_{<D}[X]$ *for every choice of* $\bar{\bar{f}}$ *made by* $\mathrm{P}_{\mathsf{poly}}$ *when following the protocol correctly.*

4. *After receiving the answers from* $\mathscr{I}$ *regarding the identities,* $\mathrm{V}_{\mathsf{poly}}$ *outputs* acc *if all identities hold, and outputs* rej *otherwise.*

As in [GWC19], we define polynomial protocols for relations in the natural way.

Definition 6.2. *Given a relation* $\mathscr{R}$*, a* polynomial protocol for $\mathscr{R}$ *is a polynomial protocol with the following additional properties.*

1. *At the beginning of the protocol,* $\mathrm{P}_{\mathsf{poly}}$ *and* $\mathrm{V}_{\mathsf{poly}}$ *are both additionally given an input* x*. The description of* $\mathrm{P}_{\mathsf{poly}}$ *assumes possession of* ω *such that* $(\mathsf{x}, \omega) \in \mathscr{R}$*.*

2. ***Completeness:*** *If* $\mathrm{P}_{\mathsf{poly}}$ *follows the protocol correctly using a witness* ω *for* x*,* $\mathrm{V}_{\mathsf{poly}}$ *accepts with probability one.*

3. ***Knowledge Soundness:*** *There exists an efficient* E*, that given access to the messages of* $\mathrm{P}_{\mathsf{poly}}$ *to* $\mathscr{I}$ *outputs* ω *such that, for any strategy of* $\mathrm{P}_{\mathsf{poly}}$*, the probability of the following event is* $\mathsf{negl}(\lambda)$*.*

   (a) $\mathrm{V}_{\mathsf{poly}}$ *outputs* acc *at the end of the protocol, and*

   (b) $(\mathsf{x}, \omega) \notin \mathscr{R}$*.*

Remark 6.3. *At this point in [GWC19] we defined a further abstraction of polynomial protocols on ranges and showed a reduction from them to polynomial protocols. We do not do this here, as this reduction from [GWC19] adds a round to the protocol, which, as already mentioned above, hurts verifier efficiency which is the focus of this paper.*

6.1 From polynomial protocols to protocols against algebraic adversaries

We wish to use the polynomial commitment scheme of Section 5.1 to compile a polynomial protocol into one with knowledge soundness in the algebraic group model.

For the purpose of capturing the efficiency of the transformation, we first define somewhat technical measures of a $(d, D, r, T, \bar{t}, \mathbf{S})$-polynomial protocol $\mathscr{P}$.

Let M^{**} be the number of distinct polynomials $\{h_{i,j}(v_{i,j}(X))\}$ appearing in the protocol in the idenitites $G_i(X, h_{i,1}(v_{i,1}(X)), \ldots, h_{i,M}(v_{i,M}(X))) \equiv 0$ checked by $\mathrm{V}_{\mathsf{poly}}$ in $\mathscr{P}$. Let $M^* = M^{**} - K$, where K is the number of identities such that G_i is linear in X_M. For $i \in [r]$, suppose $n_i = \max_{j \in [<t_i]} [t_i \cdot \deg(f_{i,j}) + i]$, where the maximum is over $\bar{f}_i$ sent by the honest prover in round i.

Let $n(\mathscr{P}) := \max_i [n_i]$. Finally, define $\mathsf{e}(\mathscr{P}) := \sum_{i<r} n_i + 2n(\mathscr{P}) + r$.

Lemma 6.4. *Let* $\mathscr{P}$ *be a* $(d, D, r, T, \bar{t}, \mathbf{S})$*-polynomial protocol over* $\mathbb{F}$ *for a relation* $\mathscr{R}$*, where*

- $T = \{t \leq A |\ t|A\}$ *for a constant* A *dividing* $p-1$.
- $\mathbf{S}$ *is the set of* A*'th powers in* $\mathbb{F}$.

Then we can construct a protocol $\mathscr{P}^*$ *for* $\mathscr{R}$ *with knowledge soundness in the Algebraic Group Model under* $2n(\mathscr{P})$*-DLOG such that*

1. *The prover* $\mathbf{P}$ *in* $\mathscr{P}^*$ *requires* $\mathsf{e}(\mathscr{P})$ $\mathbb{G}_1$*-exponentiations.*
2. *The total prover communication consists of* $r+2$ $\mathbb{G}_1$ *elements and* M^* $\mathbb{F}$*-elements.*
3. *The verifier* $\mathbf{V}$ *requires* $r+4$ $\mathbb{G}_1$*-exponentiations, two pairings, one evaluation of each* G_i *checked in* $\mathscr{P}$*, and one evaluation of each* $v_{i,j}$.

Proof. Let $\mathscr{S} = (\mathsf{gen}, \mathsf{com}, \mathsf{open})$ be the $(T, \mathbf{S})$-polynomial commitment scheme described in Theorem 5.2. Let $\bar{g} := (g_1, \ldots, g_\ell)$. The SRS of $\mathscr{P}^*$ includes $\mathsf{srs} = \mathsf{gen}(d)$, with the addition of $\mathsf{com}(\bar{g})$.

Given $\mathscr{P}$ we describe $\mathscr{P}^*$. $\mathbf{P}$ and $\mathbf{V}$ behave identically to $\mathrm{P}_{\mathsf{poly}}$ and $\mathrm{V}_{\mathsf{poly}}$, except the following changes

- As preprocessing, we compute the commitment $\mathsf{com}(t_0, \bar{f}_0)$ to the preprocessed polynomials and give this in advance to $\mathbf{V}$.
- If in round i of $\mathscr{P}$, $\mathrm{P}_{\mathsf{poly}}$ sends the vector of polynomials $\bar{f}_i \in (\mathbb{F}_{<d}[X])^{t_i}$ to $\mathscr{I}$, in $\mathscr{P}^*$ $\mathbf{P}$ sends $\mathsf{cm}_i = \mathsf{com}(t_i, \bar{f}_i)$ to $\mathbf{V}$.
- When $\mathrm{V}_{\mathsf{poly}}$ asks in $\mathscr{P}$ about the k identities

$$F_i(X) := G_i(X, h_{i,1}(v_{i,1}(X)), \ldots, h_{i,M}(v_{i,M}(X))) \equiv 0,$$

  1. Let $v_1^*, \ldots, v_{t^*}^*$ be the distinct polynomials amongst $\{v_{i,j}\}$ among the different identities.
  2. $\mathbf{V}$ chooses random $x \in \mathbf{S}$, computes $v_1^*(x), \ldots, v_{t^*}^*(x)$, and sends x to $\mathbf{P}$.
  3. $\mathbf{P}$ generally replies with $\{s_{i,j}\}_{i\in[k],j\in[M]}$, which are the alleged values $\{h_{i,j}(v_{i,j}(x)\}$. Note though that when G_i is linear in X_M, there is no need to send the value $s_{i,M}$ as the unique value that will cause the equation to be satisfied can be computed by $\mathrm{V}_{\mathsf{poly}}$ herself.
  4. $\mathbf{V}$ engages in the protocol open with $\mathbf{P}$ to verify the correctness of $\{s_{i,j}\}$
  5. $\mathbf{V}$ outputs acc if and only if for each $i \in [k]$

$$G_i(x, s_{i,1}, \ldots, s_{i,M}) = 0.$$

The efficiency claims about $\mathscr{P}^*$ follow directly from Theorem 5.2.

To prove the claim about knowledge soundness in the AGM we must describe the extractor E for the protocol $\mathscr{P}^*$. For this purpose, let $E_{\mathscr{P}}$ be the extractor of the protocol $\mathscr{P}$ as guaranteed to exist from Definition 6.2, and $E_{\mathscr{S}}$ be the extractor for the Knowledge Soundness game of $\mathscr{S}$ as in Definition 4.1.

Now assume an algebraic adversary $\mathscr{A}$ is taking the role of $\mathbf{P}$ in $\mathscr{P}^*$.

1. E sends the commitments $\overline{\mathsf{cm}}$ to $E_{\mathscr{S}}$ and receives in return $\bar{\bar{f}} \in (\mathbb{F}_{<d}[X])^{(2)}$.
2. E plays the role of $\mathscr{I}$ in interaction with $E_{\mathscr{P}}$, sending him the polynomials $\bar{\bar{f}}$.
3. When $E_{\mathscr{P}}$ outputs ω, E also outputs ω.

Now let us define two events (over the randomness of $\mathbf{V}$,$\mathscr{A}$ and gen):

1. We think of an adversary $\mathscr{A}_{\mathscr{P}}$ participating in $\mathscr{P}$, and using the polynomials $\bar{\bar{f}}$ as their messages to $\mathscr{I}$. We define A to be the event that one of the identities F_i held, but $(\mathsf{x}, \omega) \notin \mathscr{R}$. By the KS of $\mathscr{P}$, $\Pr(A) = \mathsf{negl}(\lambda)$.
2. We let B be the event that for some $i \in [k], j \in [M], h_{i,j}(v_{i,j}(x)) \neq s_{i,j}$, and at the same time V has output acc when open was run as a subroutine in Step 4. By the KS of $\mathscr{S}$, $\Pr(B) = \mathsf{negl}(\lambda)$.

Now look at the event C that $\mathbf{V}$ outputs acc, but E failed in the sense that $(\mathsf{x}, \omega) \notin \mathscr{R}$. We split C into two events.

1. A or B also happened - this has $\mathsf{negl}(\lambda)$ probability.
2. C happened but not A or B. This means that for some $i \in [k]$, F_i is not the zero polynomial, but $F_i(x) = 0$; which happens w.p. $\deg(F_i) \cdot A/p$ which is $\mathsf{negl}(\lambda)$.

□

7 Polynomial protocol for constraint system satisfiability

As in Section 6 and 7 of [GWC19], we work with a constraint system $\mathscr{C} = (\mathscr{V}, \mathscr{Q})$ where $\mathscr{Q} = (\mathbf{q_L}, \mathbf{q_R}, \mathbf{q_O}, \mathbf{q_M}, \mathbf{q_C}) \in (\mathbb{F}^n)^5$ are our "Selector vectors"; and $\mathscr{V}$ implicitly describe a permutation on $[3n]$.

We present a slightly modified polynomial protocol for the relation $\mathscr{R}_{\mathscr{C}}$ described in [GWC19]; which is the set of pairs (x, ω) with $\mathsf{x} \in \mathbb{F}^\ell, \omega \in \mathbb{F}^{m-\ell}$ such that $\mathbf{x} := (\mathsf{x}, \omega)$ satisfies $\mathscr{C}$. The difference from [GWC19] is that we do not wish to use their reduction from ranged polynomial protocols to polynomial protocols, as this adds a round of interaction, which ends up adding a verifier scalar multiplication in the compiled protocol against algebraic adversaries. Instead, we need to explicitly describe sending the quotient polynomial involved in each of three identities checked. We assume below n is a power of two such that $24n$ divides $p-1$, and $H \subset \mathbb{F}$ is a multiplicative subgroup of $\mathbb{F}$ of order n with generator $\mathbf{g}$. Note that our divisibility assumption implies $\mathbf{g}$ is a 24'th power in$\mathbb{F}$.

Preprocessed polynomials: The polynomials $\mathsf{S}_{\sigma 1}, \mathsf{S}_{\sigma 2}, \mathsf{S}_{\sigma 3}$ describing the permutation derived from $\mathscr{C}$ as in Section 8 of [GWC19]. (As explained there, the polynomials describing the identity permutation can be computed in $\log n$ time directly by the verifier.) The polynomials $\mathbf{q_L}, \mathbf{q_R}, \mathbf{q_O}, \mathbf{q_M}, \mathbf{q_C} \in \mathbb{F}_{<n}[X]$ (as in [GWC19], we identify the vectors with polynomials obtaining the vector values on H).

Protocol:

1. Let $\mathbf{x} \in \mathbb{F}^m$ be $\mathrm{P}_{\mathsf{poly}}$'s assignment consistent with the public input x. $\mathrm{P}_{\mathsf{poly}}$ computes the three polynomials $f_L, f_R, f_O \in \mathbb{F}_{<n}[X]$, where for $i \in [n]$

$$f_L(i) = \mathbf{x}_{\mathbf{a}_i}, f_R(i) = \mathbf{x}_{\mathbf{b}_i}, f_O(i) = \mathbf{x}_{\mathbf{c}_i}.$$

2. $\mathrm{P}_{\mathsf{poly}}$ and $\mathrm{V}_{\mathsf{poly}}$ compute the "Public input polynomial"

$$\mathsf{PI}(X) := \sum_{i \in [\ell]} -\mathsf{x}_i \cdot L_i(X).$$

3. $\mathrm{P}_{\mathsf{poly}}$ computes the quotient polynomial $T_0(X)$ showing f_L, f_R, f_O satisfy the arithmetic constraint; i.e.

$$T_0(X) := \frac{\mathbf{q_L}(X) \cdot f_L(X) + \mathbf{q_R}(X) \cdot f_R(X) + \mathbf{q_O}(X) \cdot f_O(X) + \mathbf{q_M}(X) \cdot f_L(X) \cdot f_R(X) + (\mathbf{q_C}(X) + \mathsf{PI}(X))}{Z_H(X)}$$

   $\mathrm{P}_{\mathsf{poly}}$ sends $\bar{f}_1 = (f_L, f_R, f_O, T_0)$ to $\mathscr{F}$.

4. $\mathrm{P}_{\mathsf{poly}}$ and $\mathrm{V}_{\mathsf{poly}}$ run an extended permutation check protocol as in [GWC19], using the permutation σ between (f_L, f_R, f_O) and itself. As explained in Section 5 of [GWC19], this exactly checks whether (f_L, f_R, f_O) copy-satisfies $\mathscr{T}_{\mathscr{C}}$. More precisely,

   (a) $\mathrm{V}_{\mathsf{poly}}$ chooses random $\beta, \gamma \in \mathbb{F}$ and sends them to $\mathrm{P}_{\mathsf{poly}}$.

   (b) Let $f'_j := f_j + \beta \cdot \mathsf{S}_{\mathsf{ID}j} + \gamma$, and $g'_j := g_j + \beta \cdot \mathsf{S}_{\sigma j} + \gamma$. That is, for $j \in \{L, R, O\}, i \in [n]$

$$f'_j(\mathbf{g}^i) = f_j(\mathbf{g}^i) + \beta((j-1) \cdot n + i) + \gamma, g'_j(\mathbf{g}^i) = g_j(\mathbf{g}^i) + \beta \cdot \sigma((j-1) \cdot n + i) + \gamma$$

   (c) Define $f', g' \in \mathbb{F}_{<3n}[X]$ by

$$f'(X) := \prod_{j \in \{L,R,O\}} f'_j(X), g'(X) := \prod_{j \in \{L,R,O\}} g'_j(X).$$

   (d) $\mathrm{P}_{\mathsf{poly}}$ computes $Z \in \mathbb{F}_{<n}[X]$, such that $Z(\mathbf{g}) = 1$; and for $i \in \{2, \ldots, n\}$

$$Z(\mathbf{g}^i) = \prod_{1 \leq \ell < i} f'(\mathbf{g}^j)/g'(\mathbf{g}^j).$$

   (e) $\mathrm{P}_{\mathsf{poly}}$ computes the quotients T_1, T_2 showing that Z "starts from one" and that Z accumulates the values of f/g. Namely,

$$T_1(X) := (L_1(X)(Z(X) - 1))/Z_H(X) = 0$$

$$T_2(X) := (Z(X)f'(X) - g'(X)Z(X \cdot \mathbf{g}))/Z_H(X)$$

(f) $\mathsf{P}_{\mathsf{poly}}$ sends $\bar{f}_2 = (Z, T_1, T_2)$ to $\mathscr{I}$.

(g) $\mathsf{V}_{\mathsf{poly}}$ checks the following three identities

i.

$$\mathbf{q_L}(X)\cdot f_L(X)+\mathbf{q_R}(X)\cdot f_R(X)+\mathbf{q_O}(X)\cdot f_O(X)+\mathbf{q_M}(X)\cdot f_L(X)\cdot f_R(X)+(\mathbf{q_C}(X)+\mathsf{PI}(X)) = T_0(X)\cdot Z_H(X)$$

ii.

$$L_1(X)(Z(X)-1) = T_1(X)Z_H(X)$$

iii.

$$Z(X)f'(X) - g'(X)Z(X\cdot\mathbf{g}) = T_2(X)Z_H(X)$$

and outputs acc iff all checks hold.

Using the analysis of [GWC19] we get

Theorem 7.1. *The above is a polynomial protocol for the relation $\mathscr{R}_{\mathscr{C}}$.*

Now using Lemma 6.4 we get

Corollary 7.2. *Assume the Q-DLOG for $Q = 18\cdot n$. Assume $24|(p-1)$. Then there is a protocol for the relation $\mathscr{R}_{\mathscr{C}}$ with Knowledge Soundness in the Algebraic Group Model such that*

1. *The prover $\mathbf{P}$ requires $35n$ $\mathbb{G}_1$-exponentiations.*
2. *The total prover communication consists of 4 $\mathbb{G}_1$-elements and 15 $\mathbb{F}$-elements.*
3. *The verifier requires 5 $\mathbb{G}_1$-exponentiations and two pairings.*

Proof. We need to simply compute the parameters we are plugging into Lemma 6.4 when using the above protocol $\mathscr{P}$. The number of rounds r is two. We have

- $n_0 = \max_{j<8}[8\cdot(n-1)+j] < 8n$
- $n_1 = \max\{4\cdot\deg(f_L), 4\cdot\deg(f_R)+1, 4\cdot\deg(f_O)+2, 4\cdot\deg(T_0)+2\} < 8n$
- $n_2 = \max\{3\cdot\deg(Z), 3\cdot\deg(T_1)+1, 3\cdot\deg(T_2)+2\} < 9n$
- $n(\mathscr{P}) = \max\{n_0, n_1, n_2\} < 9n$

Now, from Lemma 6.4 we know that

1. The prover requires $\mathsf{e}(\mathscr{P}) = n_1 + n_2 + 2n(\mathscr{P}) + r \leq 35n$ $\mathbb{G}_1$-exponentiations.
2. The total prover communication consists of $r+2 = 4$ $\mathbb{G}_1$ elements and $M^* = 15$ $\mathbb{F}$-elements.(Opening $\mathbf{q_L}, \mathbf{q_R}, \mathbf{q_O}, \mathbf{q_M}, \mathbf{q_C}, \mathsf{S}_{\sigma 1}, \mathsf{S}_{\sigma 2}, \mathsf{S}_{\sigma 3}, f_L, f_R, f_O, Z$ at x and Z, T_1, T_2 at $\mathbf{g}x$.)
3. The verifier $\mathbf{V}$ requires $r+4 = 6$ $\mathbb{G}_1$-exponentiations.

□

References

[BDFG] D. Boneh, J. Drake, B. Fisch, and A. Gabizon. Halo infinite: Proof-carrying data from additive polynomial commitments. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part I.*

[BDFG20] D. Boneh, J. Drake, B. Fisch, and A. Gabizon. Efficient polynomial commitment schemes for multiple points and polynomials. *IACR Cryptol. ePrint Arch.*, page 81, 2020.

[BFS19] B. Bünz, B. Fisch, and A. Szepieniec. Transparent snarks from DARK compilers. *IACR Cryptology ePrint Archive*, 2019:1229, 2019.

[BGM17] S. Bowe, A. Gabizon, and I. Miers. Scalable multi-party computation for zk-snark parameters in the random beacon model. Cryptology ePrint Archive, Report 2017/1050, 2017. `https://eprint.iacr.org/2017/1050`.

[But] V. Buterin. *https://vitalik.ca/general/2021/01/05/rollup.html.*

[CHM+19] A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. P. Ward. Marlin: Preprocessing zksnarks with universal and updatable SRS. *IACR Cryptology ePrint Archive*, 2019:1047, 2019.

[Gab19] A. Gabizon. Auroralight:improved prover efficiency and SRS size in a sonic-like system. *IACR Cryptology ePrint Archive*, 2019:601, 2019.

[GKM+] J. Groth, M. Kohlweiss, M. Maller, S. Meiklejohn, and I. Miers. Updatable and universal common reference strings with applications to zk-snarks. *IACR Cryptology ePrint Archive*, 2018.

[GWC19] A. Gabizon, Z. J. Williamson, and O. Ciobotaru. PLONK: permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. *IACR Cryptology ePrint Archive*, 2019:953, 2019.

[KZG10] A. Kate, G. M. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, pages 177–194, 2010.

[MBKM19] 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*, 2019:99, 2019.