Polynomial Commitments

This is an approach to vector commitments via polynomial coefficients. We have used the KZG10 polynomial commitment scheme throughout this write-up, although other polynomial commitment schemes might also work at the cost of larger proof sizes.

This is different from the more commonly used approach of committing to the polynomial (computed via Lagrange interpolation) valued v_i at ω^i for a fixed root of unity ω. Our scheme supports features compatibility with linear transformations and permutations etc. that do not seem to be supported by the Lagrange interpolation-based Vector commitment. Furthermore, this scheme does not require the existence of a large smooth-order multiplicative subgroup in the scalar field, which has been shown to be an attack vector ([Cheon10]).

Our approach is compatible with both PLONK and the R1CS Snarks. We have described a variant of PLONK via this approach in Chapter 5. It supports arbitrary polynomial custom gates and does not need a a large smooth-order multiplicative subgroup in the scalar field.

We further note that our approach is potentially useful for recursive Snarks. The fact that we do not require large 2-power roots of unity to lie in the fields opens up the possibility of using a wider class of amicable curve pairs for recursion.

The protocols and the relations they are HVZKs for are listed on the last page for easy reference. If nothing else, we hope the protocols developed in this write-up shed light on some mildly interesting properties of univariate polynomials over finite fields.

Remark: I have made sure that the proof sizes and verification times are constant but haven't yet put in more effort into optimizing the proof sizes. Most of the proof sizes can be made smaller by making the protocols less modular and by sharing challenges for various subprotocols.

0.1 A Vector commitment

We define a simple vector commitment using the polynomial commitment scheme. A vector $\mathbf{v} = (v_0, \cdots, v_n) \in \mathbb{F}_p^{n+1}$ is identified with the polynomial $\sum_{i=0}^{n} v_i \cdot X^i$, which is then committed as in [KZG10]. Thus, for a vector $\mathbf{v} = (v_0, \cdots, v_n) \in \mathbb{F}_p^{n+1}$, we define the commitment

$$\texttt{Com}(\mathbf{v}) \ := \ g_1^{\sum_{i=0}^{n} v_i \cdot s^i} \ = \ \prod_{i=0}^{n} (g_1^{s^i})^{v_i} \ \in \ \mathbb{G}_1 \ ,$$

where s is the trapdoor.

In this paper, we will show that this vector commitment supports zero-knowledge proofs of *constant size* and with *constant verification runtime* for all of the following:

• a committed vector being the Hadamard (aka entry-wise) product of two other committed vectors

• a batched membership proof, i.e. a proof of two committed vectors agreeing at every index in a hidden committed index set

• a batched non-membership/disjointness proof, i.e. a proof of two committed vectors disagreeing at every index in a hidden committed index set

• a proof of non-repetition, i.e. no element in the committed vector appearing more than once in this vector. This easily generalizes to a proof that no element appears more than k times.

• a committed vector being a permutation of another, for some secret permutation known to the Prover

• two committed vectors $\mathbf{v}$, $\mathbf{w}$ and two (hidden) committed index sets $\mathcal{I}$, $\mathcal{J}$ being such that the sequences $(v_i)_{i\in\mathcal{I}}$, $(w_i)_{i\in\mathcal{J}}$ coincide; a special case of this is one committed vector being a subsequence of another committed vector

• two committed vectors having the same underlying set (possibly with different multiplicities of elements)

• the *subset-sum problem*, i.e. some of the entries of a committed vector adding up to a certain total value (which can be a public value or a hidden committed value)

• the *knapsack problem*, which generalizes the subset sum problem by allowing for weights.

• a proof linking the commitment to the vector to a commitment to the size of the committed data.

0.2 The setup

Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be cyclic groups of order p for some prime p such that there exists a pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ which is *bilinear*, *non-degenerate* and *efficiently computable*. Fix generators g_1, g_2 of the cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$ respectively. Then $\mathbf{e}(g_1, g_2)$ is a generator of $\mathbb{G}_T$. For a trapdoor $s \in \mathbb{F}_p^*$, the common reference string (CRS) is given by

$$[g_1, g_1^s, \cdots, g_1^{s^n}] \ , \ [g_2, g_2^s, \cdots, g_2^{s^n}]$$

The verification key is $[g_1, g_1^s], [g_2, g_2^s]$. The Verifier just needs these four points for any verification. These are computed via a one-time secure MPC.

In practice, it would be suitable to use the optimal ate pairing with a pairing-friendly ordinary curve E over a finite prime field $\mathbb{F}_\ell$ such that:

- $|E(\mathbb{F}_\ell)|$ has a prime divisor p of bit-size ≈ 256

- E has embedding degree $k \leq 24$ with respect to p

- the prime power ℓ^k is roughly of bit-size 4500.

Unlike the pairing-based schemes that commit to vectors via values at roots of unity, our scheme does not need the scalar field $\mathbb{F}_p$ to have a large smooth-order multiplicative subgroup. Although this isn't an avenue we have explored yet, this is potentially useful for recursive pairing-based schemes since it opens up the possibility of using amicable curve pairs where the prime fields do not have large smooth-order multiplicative subgroups.

The lack of 2-power roots of unity in the scalar field causes a slowdown when it comes to the FFTs (and consequently the Prover's runtime). However, recent techniques such as [BCKL21] have vastly sped up the FFTs in such finite fields and the slowdown is substantially better than the prior $\log(\log(n))$ multiplicative factor. Furthermore, while our scheme requires the Prover to use FFTs for polynomial multiplications, it does not require polynomial divisions with divisors of degree larger than one.

Hiding commitments: Fix integers d, N such that for any committed polynomial $f(X)$, $N > d > 2 \cdot \deg(f)$. A hiding polynomial commitment is given by

$$\texttt{HCom}(f(X)) := g_1^{f(s)+s^d \cdot h(s)}$$

for a randomly generated $h(X) \in \mathbb{F}_p[X]_{<d}$. Thus, the integers d, N have to large enough so that d and $N-d$ are larger than the degree of any polynomial to be committed.

For simplicity and brevity, we ignore the blinding factor $g_1^{s^d \cdot h(s)}$ and use the binding (non-hiding) commitment $g_1^{f(s)}$ in the ZK protocols in this write-up. But the protocols can be made genuinely zero-knowledge by adding such blinding factors and subsequently providing proofs of knowledge of the exponent with base $g_1^{s^d}$.

0.3 A commitment to a index set

Let $I \subseteq [\text{length}(\texttt{CRS})]$ be a index set. We commit to the set $\mathcal{I}$ by committing to the binary polynomial

$$\chi_{\mathcal{I}}(X) := \sum_{i \in \mathcal{I}} X^i.$$

Thus, the commitment is given by

$$\texttt{Com}(\mathcal{I}) := \texttt{Com}(\chi_{\mathcal{I}}(X)) = g_1^{\chi_{\mathcal{I}}(s)} = \prod_{i \in \mathcal{I}} g_1^{\sum_{i \in \mathcal{I}} s^i}$$

We call the polynomial $\chi_{\mathcal{I}}(X)$ the *characteristic polynomial* of the index set $\mathcal{I}$.

This commitment is *dynamic* in the sense that for any update to $\mathcal{I}$, the commitment can be updated in constant runtime. Clearly, for any subset $I \subseteq [\text{length}(\texttt{CRS})]$, the polynomial $\chi_{\mathcal{I}}(X)$ is binary in the sense that every coefficient lies in $\{0,1\} \subseteq \mathbb{F}_p$. Conversely, every binary polynomial of degree n is of the form $\chi_{\mathcal{I}}(X)$ for some subset $I \subseteq [n]$

To show that an element a of $\mathbb{G}_1$ is a commitment to some known index set, it suffices to show that $a = g_1^{f(s)}$ for some binary polynomial $f(X)$. This can be done using the protocol `ZKPoBinary`.

We note a few properties of this commitment.

- The size of the index set $\mathcal{I}$ coincides with the evaluation of $\chi_{\mathcal{I}}(X)$ at $X = 1$, i.e. $|\mathcal{I}| = \chi_{\mathcal{I}}(1)$.

- The degree of the polynomial $\chi_{\mathcal{I}}(X)$ is the largest element in $\mathcal{I}$.

- The smallest element in $\mathcal{I}$ is $\text{val}_{(X)}(\chi_{\mathcal{I}}(X))$, i.e. the largest integer k such that X^k divides $\chi_{\mathcal{I}}(X)$.

Thus, these attributes of $\mathcal{I}$ are fairly straightforward to prove in zero-knowledge. Let $\odot$ denote the Hadamard (aka entry-wise) product of two polynomials (or vectors). Index sets $\mathcal{I}$, $\mathcal{J}$ satisfy the following properties:

- $\chi_{\mathcal{I} \cap \mathcal{J}}(X) = \chi_{\mathcal{I}}(X) \odot \chi_{\mathcal{J}}(X)$.
- $\chi_{\mathcal{I} \cup \mathcal{J}}(X) = \chi_{\mathcal{I}}(X) + \chi_{\mathcal{J}}(X) - \chi_{\mathcal{I} \cap \mathcal{J}}(X)$.

0.4 Older Protocols

These protcols and their security proofs are in [Th19]. We state them here because we will need them as subprotcols in the subsequent sections.

Protocol 0.1. *Proof of knowledge of the exponent with base g_1* ($\texttt{PoKE}^*$):
Parameters: A pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$; generators $g_1,\ g_2$ for $\mathbb{G}_1,\ \mathbb{G}_2$ respectively. generators $g_1,\ g_2$ for $\mathbb{G}_1,\ \mathbb{G}_2$ respectively.
Inputs: Element $a \in \mathbb{G}_1$
Claim: The Prover knows a polynomial $f(X) \in \mathbb{F}_p[X]$ such that $g_1^{f(s)} = a$.

1. The Fiat-Shamir heuristic generates a challenge $\alpha \in \mathbb{F}_p^*$.
2. $\mathcal{P}$ computes the polynomial $h(X) \in \mathbb{F}_p[X]$ and the element $\beta \in \mathbb{F}_p$ such that

$$f(X) = (X + \alpha) \cdot h(X) + \beta.$$

$\mathcal{P}$ computes $Q := g_1^{h(s)}$ and sends $(Q\ ,\ \beta) \in \mathbb{G}_1 \times \mathbb{F}_p$ to $\mathcal{V}$.
3. $\mathcal{V}$ then verifies the equations

$$\mathbf{e}(Q\ ,\ g_2^{s+\alpha}) \cdot \mathbf{e}(g_1^{\beta}\ ,\ g_2) \stackrel{?}{=} \mathbf{e}(a\ ,\ g_2)$$

and accepts if and only if both equations hold. □

Protocol 0.2. *Proof of knowledge of the exponent* ($\texttt{PoKE} - 1$):
Parameters: A pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$; generators $g_1,\ g_2$ for $\mathbb{G}_1,\ \mathbb{G}_2$ respectively.
Inputs: Elements $a, b \in \mathbb{G}_1$
Claim: The Prover knows a polynomial $f(X) \in \mathbb{F}_p[X]$ such that $a^{f(s)} = b$.

1. The Prover $\mathcal{P}$ sends $\widetilde{g}_2 := g_2^{f(s)} \in \mathbb{G}_2$ to the Verifier $\mathcal{V}$.
2. $\mathcal{P}$ sends a non-interactive proof of $\texttt{PoKE}^*[g_2,\ \widetilde{g}_2]$.
3. $\mathcal{V}$ verifies the proof of $\texttt{PoKE}^*[g_2,\ \widetilde{g}_2]$ and the equation

$$\mathbf{e}(a\ ,\ \widetilde{g}_2) \stackrel{?}{=} \mathbf{e}(b\ ,\ g_2).$$

$\mathcal{V}$ accepts if and only if the $\texttt{PoKE}^*$ is valid and the pairing equation holds. □

When the pairing is type III, the exponentiations in $\mathbb{G}_2$ are substantially more expensive than those in $\mathbb{G}_1$. Thus, in cases where the Prover knows a polynomial $e(X)$ of a *reasonably small* degree such that $a = g_1^{e(s)}$, it can be cheaper to compute the element $a^{h(s)} = g_1^{e(s)\cdot h(s)} \in \mathbb{G}_1$ instead of $g_2^{h(s)} \in \mathbb{G}_2$.

Protocol 0.3. *Proof of knowledge of the exponent* ($\texttt{PoKE} - 2$):
Parameters: A pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$; generators $g_1,\ g_2$ for $\mathbb{G}_1,\ \mathbb{G}_2$ respectively.
Inputs: Elements $a, b \in \mathbb{G}_1$
Claim: The Prover knows a polynomial $f(X) \in \mathbb{F}_p[X]$ such that $a^{f(s)} = b$.

1. The Prover $\mathcal{P}$ sends the element $\widetilde{g} := g_1^{f(s)} \in \mathbb{G}_1$.
2. The Fiat-Shamir heuristic generates a challenge $\alpha \in \mathbb{F}_p^*$.
3. $\mathcal{P}$ computes the polynomial $h(X) \in \mathbb{F}_p[X]$ and the element $\beta \in \mathbb{F}_p$ such that

$$f(X) = (X + \alpha) \cdot h(X) + \beta.$$

$\mathcal{P}$ computes

$$Q := a^{h(s)} \quad , \quad \widehat{g} := g_1^{h(s)}$$

and sends $(Q,\ \widehat{g},\ \beta) \in \mathbb{G}_1^2 \times \mathbb{F}_p^*$ to $\mathcal{V}$.

4. $\mathcal{V}$ verifies the equations

$$\mathbf{e}(Q\ ,\ g_2^{s+\alpha}) \stackrel{?}{=} \mathbf{e}(b \cdot a^{-\beta}\ ,\ g_2) \bigwedge \mathbf{e}(\widehat{g}\ ,\ g_2^{s+\alpha}) \stackrel{?}{=} \mathbf{e}(\widetilde{g} \cdot g_1^{-\beta}\ ,\ g_2)$$

and accepts if and only if both equations hold. □

Protocol 0.4. *Zero-knowledge proof of knowledge of the exponent* (`ZKPoKE`):
Parameters: A pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$; generators $g_1,\ g_2$ for $\mathbb{G}_1,\ \mathbb{G}_2$ respectively.
Inputs: Elements $a, b \in \mathbb{G}_1$
Claim: The Prover knows a polynomial $f(X) \in \mathbb{F}_p[X]$ such that $a^{f(s)} = b$.

1. The Prover $\mathcal{P}$ chooses a random $k \in \mathbb{F}_p^*$ and sends $u := a^k \in \mathbb{G}_1$ to the Verifer $\mathcal{V}$.
2. The Fiat-Shamir heuristic generates a challenge $\alpha \in \mathbb{F}_p^*$.
3. $\mathcal{P}$ sends a non-interactive proof of $\texttt{PoKE}[a,\ b^\alpha \cdot u]$.
4. $\mathcal{V}$ independently computes $b^\alpha \cdot u \in \mathbb{G}_1$ and accepts if and only if the proof of $\texttt{PoKE}[a,\ b^\alpha \cdot u]$ is valid. □

1 Some preliminary protocols

The next protocol allows a Prover to prove in zero-knowledge that given elements a, b in $\mathbb{G}_1$, he knows a constant α such that $b = a^\alpha$. This is basically Schnorr's protocol, which does not need pairings or the public parameters.

The key idea is that for polynomials $f(X)$, $h(X)$ and a randomly generated challenge $\gamma \in \mathbb{F}_p$, if the sum $f(X) \cdot \gamma + h(X)$ is a constant, then with overwhelming probability, the polynomials $f(X)$, $h(X)$ are constants as well.

Protocol 1.1. *Zero-knowledge proof of constant discrete logarithm* (`ZKPoConst`)
(aka Schnorr's protocol)
Parameters: A pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$;
Inputs: Elements $a, b \in \mathbb{G}_1$
Claim: The Prover knows a constant $\alpha \in \mathbb{F}_p$ such that $b = a^\alpha$

1. The Prover $\mathcal{P}$ chooses a random blinding scalar $\beta \in \mathbb{F}_p$ and sends $C_\beta := a^\beta$.
2. The Fiat-Shamir heuristic generates a challenge γ.
3. The Prover sends $m := \beta + \alpha \cdot \gamma$.
4. $\mathcal{V}$ accepts if and only if $m \in \mathbb{F}_p$ and $a^m = C_\beta \cdot b^\gamma$. □

In case the Prover needs to show that the exponent α is *non-zero*, he could combine this protocol with a proof that he knows the discrete logarithm α^{-1} between b and a. We next

discuss the Chaum-Pedersen protocol in the bilinear pairing setting. As with Schnorr's protocol, it applies to the bilinear setting without any modifications.

Protocol 1.2. *Zero-knowledge proof of equality of constant discrete logarithms*
(aka the Chaum-Pedersen protocol)
Parameters: A pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$;
Inputs: Elements $a_1, b_1, a_2, b_2 \in \mathbb{G}_1$
Claim: The Prover knows a constant $\alpha \in \mathbb{F}_p$ such that $b_1 = a_1^\alpha$, $b_2 = a_2^\alpha$.

1. The Prover $\mathcal{P}$ chooses a random blinding scalar $\beta \in \mathbb{F}_p$ and sends
$$C_{\beta,1} := a_1^\beta \ \ , \ \ C_{\beta,2} := a_2^\beta$$
2. The Fiat-Shamir heuristic generates a challenge γ.
3. The Prover sends $m := \beta + \alpha \cdot \gamma$.
4. $\mathcal{V}$ accepts if and only if $m \in \mathbb{F}_p$ and the equations
$$a_1^m = C_{\beta,1} \cdot b_1^\gamma \ \ , \ \ a_2^m = C_{\beta,2} \cdot b_2^\gamma$$
hold. □

In case the Prover needs to show that the exponent α is *non-zero*, he could combine this protocol with a proof that he knows the discrete logarithm α^{-1} between b and a.

The next protocol allows a Prover to show that it is the same polynomial committed with two distinct bases in $\mathbb{G}_1$.

Protocol 1.3. *Equality of polynomial exponents* (`ZKPoEq`)
Parameters: A pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$; generators $g_1,\ g_2$ for $\mathbb{G}_1,\ \mathbb{G}_2$ respectively.
Inputs: Elements $a_1, a_2, b_1, b_2 \in \mathbb{G}_1$
Claim: The Prover knows a polynomial $f(X)$ such that $b_1 = a_1^{f(s)}$, $b_2 = a_2^{f(s)}$

1. The Prover $\mathcal{P}$ sends $\widetilde{g}_2 := g_2^{f(s)} \in \mathbb{G}_2$ along with a proof of `ZKPoKE`$[g_2,\ \widetilde{g}_2]$.
2. The Verifier $\mathcal{V}$ verifies the `ZKPoKE` and the (batchable) equations
$$\mathbf{e}(a_1\ ,\ \widetilde{g}_2) \stackrel{?}{=} \mathbf{e}(b_1\ ,\ g_2) \ \ , \ \ \mathbf{e}(a_2\ ,\ \widetilde{g}_2) \stackrel{?}{=} \mathbf{e}(b_2\ ,\ g_2)$$
□

The next protocol opens a polynomial commitment in zero knowledge. Given elements $a, a_1, b_1 \in \mathbb{G}_1$, a Prover can use the protocol to prove in zero-knowledge that he knows a polynomial $f(X) \in \mathbb{F}_p[X]$ and a constant $\alpha \in \mathbb{F}_p$ such that
$$g_1^{f(s)} = a \ \ , \ \ g_1^\alpha = a_1 \ \ , \ \ g_1^{f(\alpha)} = b_1.$$
The Prover first proves in zero-knowledge that he knows *constants* α, β such that between $g_1^\alpha = a_1$ and $g_1^\beta = b_1$. He then proves in zero-knowledge that $f(X) - \beta$ is divisible by $X - \alpha$, which implies that $\beta = f(\alpha)$.

Protocol 1.4. *ZK opening of polynomial commitment* (`ZKOpen`)
Parameters: A pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$; generators $g_1,\ g_2$ for $\mathbb{G}_1,\ \mathbb{G}_2$ respectively.
Inputs: Elements $a, a_1, b_1 \in \mathbb{G}_1$
Claim: The Prover knows a polynomial $f(X) \in \mathbb{F}_p[X]$ and a constant $\alpha \in \mathbb{F}_p$ such that

$$g_1^{f(s)} = a \ \ , \ \ g_1^{\alpha} = a_1 \ \ , \ \ g_1^{f(\alpha)} = b_1.$$

1. The Prover $\mathcal{P}$ sends a proof of $\texttt{ZKPoKE}^*[g_1,\ a]$.
2. $\mathcal{P}$ sends (batchable) proofs for $\texttt{ZKPoConst}[g_1,\ a_1]$, $\texttt{ZKPoConst}[g_1,\ b_1]$.
3. $\mathcal{P}$ sends a proof of $\texttt{ZKPoKE}[g_1^s \cdot a_1^{-1},\ a \cdot b_1^{-1}]$.
4. $\mathcal{V}$ accepts if and only if the `ZKPoKE`s and the `ZKPoConst`s are valid. □

Given three polynomial commitments, the next protocol allows a Prover to prove in zero-knowledge that the third polynomial is the product of the first two. This is an HVZK argument of knowledge for the following relation:

$$\mathcal{R}_{\texttt{Prod}}[g_1,\ (a_1, a_2),\ a_{1,2}] = \left\{ \begin{array}{l} \big((a_1, a_2 \in \mathbb{G}_1),\ f_1(X),\ f_2(X)\ \in\ \mathbb{F}_p[X]\big) : \\ g_1^{f_1(s)} = a_1\ ,\ g_1^{f_2(s)} = a_2\ ,\ g_1^{f_1(s)\cdot f_2(s)} = a_{1,2} \end{array} \right\}$$

Protocol 1.5. *Zero-knowledge proof of product* (`ZKPoProd`)
Parameters: A pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$; generators $g_1,\ g_2$ for $\mathbb{G}_1,\ \mathbb{G}_2$ respectively.
Inputs: Elements $a_1, a_2, a_{1,2} \in \mathbb{G}_1$
Claim: The Prover knows polynomials $f_1(X), f_2(X) \in \mathbb{F}_p[X]$ such that

$$g_1^{f_1(s)} = a_1 \ \ , \ \ g_1^{f_2(s)} = a_2 \ \ , \ \ g_1^{f_1(s)\cdot f_2(s)} = a_{1,2}.$$

1. The Prover $\mathcal{P}$ sends (batchable) proofs for $\texttt{ZKPoKE}^*[g_1,\ a_1]$, $\texttt{ZKPoKE}^*[g_1,\ a_2]$.
2. $\mathcal{P}$ sends $b_2 := g_2^{f_2(s)} \in \mathbb{G}_2$.
3. $\mathcal{V}$ verifies the `ZKPoKE`*s and the (batchable) equations

$$\mathbf{e}(g_1\ ,\ b_2) \stackrel{?}{=} \mathbf{e}(a_2\ ,\ g_2) \ \ , \ \ \mathbf{e}(a_1\ ,\ b_2) \stackrel{?}{=} \mathbf{e}(a_{1,2}\ ,\ g_2)$$

and accepts if and only if they are all valid. □

The next protocol is fairly simple but will be needed repeatedly for subsequent protocols. Given a committment to a polynomial $f(X)$, we discuss a protocol that allows a Prover to verifiably send a commitment to the twist $f(\gamma \cdot X)$ for a publicly known element $\gamma \in \mathbb{F}_p$. It hinges on the observation that for a random element $\beta \in \mathbb{F}_p$, if $f_1(X) \equiv f(\beta) \pmod{(\gamma \cdot X - \beta)}$, then with overwhelming probability, $f_1(X) = f(\gamma \cdot X)$.

In response to a challenge β, the Prover sends a commitment C_β to $f(\beta)$ along with a proof (Schnorr's protocol) that this is a commitment to a known *constant*. He then provides $(s-\beta)$-th and $(\gamma \cdot s - \beta)$-th roots of $a \cdot C_\beta^{-1}$ and $a_\gamma \cdot C_\beta^{-1}$ respectively to show that:

- C_β is a commitment to $f(\beta)$

- a_γ is a commitment to a polynomial $f_1(X)$ such that $f_1(X) \equiv f(\beta) \pmod{\gamma \cdot X - \beta}$.

This implies that with overwhelming probability, $a_\gamma = g_1^{f(\gamma \cdot s)}$. This is a HVZK for the following relation:

$$\mathcal{R}_{\texttt{Twist}}[g_1,\ (a,\gamma),\ a_\gamma] = \left\{ \begin{array}{l} \big((a, a_\gamma \in \mathbb{G}_1\ ,\ \gamma \in \mathbb{F}_p),\ f(X) \in \mathbb{F}_p[X]\big) : \\ g_1^{f(s)} = a\ ,\ g_1^{f(\gamma \cdot s)} = a_\gamma \end{array} \right\}$$

Protocol 1.6. *Zero-knowledge proof of twist* (`ZKPoTwist`)
Parameters: A pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$; generators $g_1,\ g_2$ for $\mathbb{G}_1,\ \mathbb{G}_2$ respectively.
Inputs: Elements $a, a_\gamma \in \mathbb{G}_1$, $\gamma \in \mathbb{F}_p$
Claim: The Prover knows a polynomial $f \in \mathbb{F}_p[X]$ such that

$$g_1^{f(s)} = a\ \ ,\ \ g_1^{f(\gamma \cdot s)} = a_\gamma.$$

1. The Prover $\mathcal{P}$ sends a proof for $\texttt{ZKPoKE}^*[g_1,\ a]$.
2. The Fiat-Shamir heuristic generates a challenge β.
3. $\mathcal{P}$ sends the element $C_\beta := g_1^{f(\beta)}$ along with a proof for $\texttt{ZKPoConst}[g_1\ ,\ C_\beta]$.
4. $\mathcal{P}$ sends proofs for $\texttt{ZKPoKE}[g_1^{s-\beta},\ a \cdot C_\beta^{-1}]$ and $\texttt{ZKPoKE}[g_1^{s-\gamma\cdot\beta},\ a_\gamma \cdot C_\beta^{-1}]$.
5. The Verifier $\mathcal{V}$ verifies the `ZKPoKE`s and the `ZKPoConst` and accepts if and only if they are all valid. □

The next protocol demonstrates that given three polynomial commitments, the third equals composition of the first two. Given polynomials $f_1(X)$, $f_2(X)$, $f_{1,2}(X)$, the following are equivalent with overwhelming probability:

1. $f_{1,2}(X) = f_1(f_2(X))$.
2. For a random challenge $\gamma \in \mathbb{F}_p$, $f_{1,2}(X) - f_1(\gamma)$ is divisible by $f_2(X) - \gamma$.

To this end, the Prover sends a commitment to $f_1(\gamma)$ and proves that it is, in fact, a commitment to $f_1(\gamma)$. This subprotocol hinges on the fact that for a *constant* $\beta \in \mathbb{F}_p$, the following are equivalent:

1. $\beta = f_2(\gamma)$
2. $f_2(X) - \beta$ is divisible by $X - \gamma$.

This is a HVZK for the following relation:

$$\mathcal{R}_{\texttt{Comp}}[g_1,\ (a_1,a_2),\ a_{1,2}] = \left\{ \begin{array}{l} \big((a_1, a_2, a_{1,2} \in \mathbb{G}_1),\ f_1(X),\ f_2(X) \in \mathbb{F}_p[X]\big) : \\ g_1^{f_1(s)} = a_1\ ,\ g_1^{f_2(s)} = a_2\ ,\ g_1^{f_1(f_2(s))} = a_{1,2} \end{array} \right\}$$

Protocol 1.7. *Zero-knowledge proof of composition* (`ZKPoComp`)
Parameters: A pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$; generators $g_1,\ g_2$ for $\mathbb{G}_1,\ \mathbb{G}_2$ respectively.
Inputs: Elements $a_1,\ a_2,\ a_{1,2} \in \mathbb{G}_1$

Claim: The Prover knows polynomials $f_1(X),\ f_2(X)\ \in\ \mathbb{F}_p[X]$ such that

$$g_1^{f_1(s)} = a_1\ \ ,\ \ g_1^{f_2(s)} = a_2\ \ ,\ \ g_1^{f_1(f_2(s))} = a_{1,2}$$

1. The Fiat-Shamir heuristic generates a challenge γ.
2. The Prover $\mathcal{P}$ sends

$$C_{1,\gamma} := g_1^{f_1(\gamma)}\ ,\ C_{2,\gamma} := g_1^{f_2(\gamma)}\ ,\ C_{1,2,\gamma} := g_1^{f_1(f_2(\gamma))}$$

along with (batchable) proofs for $\texttt{ZKPoConst}[g_1,\ C_{1,\gamma}]$, $\texttt{ZKPoConst}[g_1,\ C_{2,\gamma}]$ and $\texttt{ZKPoConst}[g_1,\ C_{1,2,\gamma}]$

3. $\mathcal{P}$ sends the elements

$$Q_1 := g_1^{[f_1(s)-f(\gamma)]/[s-\gamma]}\ \ ,\ \ Q_2 := g_1^{[f_2(s)-f_2(\gamma)]/[s-\gamma]}\ \ ,\ \ Q_{1,2} := g_1^{[f_{1,2}(s)-f_{1,2}(\gamma)]/[s-\gamma]}$$

4. $\mathcal{P}$ sends the element $\widehat{Q} := g_1^{[f_{1,2}(s)-f_1(\gamma)]/[f_2(s)-\gamma]}$.
5. $\mathcal{V}$ accepts if and only the $\texttt{ZKPoConst}$s are valid and the (batchable) equations

$$\mathbf{e}(Q_1\ ,\ g_1^{s-\gamma}) \stackrel{?}{=} \mathbf{e}(a_1 \cdot C_{1,\gamma}^{-1}\ ,\ g_2)\ \ ,\ \ \mathbf{e}(Q_2\ ,\ g_1^{s-\gamma}) \stackrel{?}{=} \mathbf{e}(a_1 \cdot C_{2,\gamma}^{-1}\ ,\ g_2),$$

$$\mathbf{e}(Q_{1,2}\ ,\ g_1^{s-\gamma}) \stackrel{?}{=} \mathbf{e}(a_{1,2} \cdot C_{1,2,\gamma}^{-1}\ ,\ g_2)\ \ ,\ \ \mathbf{e}(\widehat{Q}\ ,\ a_2 \cdot g_2^{-\gamma}) \stackrel{?}{=} \mathbf{e}(a_{1,2} \cdot C_{1,\gamma}^{-1}\ ,\ g_2)$$

hold. □

2 Linking a committed polynomial to a commitment to its degree

In this section, we describe a protocol (and related protocols) that allows a Prover to show that a committed polynomial has degree that coincides with a committed integer. In particular, this can be used to show that two committed polynomials are of the same degree without revealing this common degree.

We briefly describe the approach. We first deal with the special case where the committed polynomial $f(X)$ is a monomial, i.e. $f(X) = c_n \cdot X^n$ for some $c_n \in \mathbb{F}_p^*$, $n \in \mathbb{Z}^{\geq 1}$. Note that

$$\deg(f(X)) = n = f'(1) \cdot f(1)^{-1}.$$

Thus, given commitments $a = g_1^{f(s)}$, $a_{\deg} = g_1^{\deg(f)}$, the Prover can verifiably send a commitment $a' = g_1^{f'(s)}$ to the derivative $f'(X)$ and then show that the the polynomials $f(X)$, $f'(X)$ are such that $f'(1) \cdot f(1)^{-1}$ is the field element committed in $a_{\deg}$.

The more general case of an arbitrary polynomial uses the special case of a monomial in conjuction with a protocol that shows that the degree of the polynomial $f_1(X) := f(X) - c_n \cdot X^n$ is less than n. Note that a polynomial $f_1(X)$ is of degree less than n is and only if the rational function $c_n X^n \cdot f_1(X^{-1})$ is a polynomial divisible by X.

The Prover first shows that $g_1^{c_n \cdot s^n}$ is a commitment to a monomial of degree n where n is the field element committed in $a_{\deg} = g_1^n$. He then sends the element $a_1 := g_1^{c_n s^n \cdot f_1(s^{-1})}$ and proves that this is a commitment to $X^n \cdot f_1(X^{-1})$. This can be done by showing that for a randomly generated challenge β, the polynomial $h(X)$ committed in a_1 is such that:

1. $h(X) - X^n \cdot f_1(\beta^{-1})$ is divisible by $X - \beta^{-1}$.
2. $h(X)$ is divisible by X.

2.1 Linking a committed polynomial to a commitment to its derivative

For a polynomial $f(X) = \sum_{i=0}^{\deg(f)} c_i \cdot X^i$, the derivative $f'(X)$ is the polynomial

$$f'(X) = \sum_{i=1}^{\deg(f)} (c_i \cdot i) \cdot X^{i-1}.$$

The next protocol allows a Prover to prove succinctly and in zero-knowledge that a committed polynomial is the derivative of another committed polynomial. It relies on the simple fact that for a polynomial $f(X)$, its derivative $f'(X)$ and any field element $\alpha \in \mathbb{F}_p$,

$$f(X) \equiv f'(\alpha) \cdot (X - \alpha) + \beta \pmod{(X-\alpha)^2}$$

for some $\beta \in \mathbb{F}_p$. In fact, the derivative is the *unique* polynomial with this property. This alternate definition will be more useful for our protocols than the standard definition .

Definition 2.1. For a polynomial $f(X)$, *the derivative* of $f(X)$ is the unique polynomial $h(X)$ such that for any element $\alpha \in \mathbb{F}_p[X]$, there exists an element $\beta \in \mathbb{F}_p$ and a polynomial $q(X) \in \mathbb{F}_p[X]$ such that

$$f(X) = q(X) \cdot (X - \alpha)^2 + h(\alpha) \cdot (X - \alpha) + \beta. \qquad \square$$

We now describe the protocol. Given commitments $a = g_1^{f(s)}$, $b = g_1^{f'(s)}$, the Fiat-Shamir heuristic generates a challenge α. The Prover then shows he knows a polynomial $q(X)$ and and an element $\beta \in \mathbb{F}_p$ such that

$$f(X) = q(X) \cdot (X - \alpha)^2 + f'(\alpha) \cdot (X - \alpha) + \beta.$$

To this end, he sends the element $C_{f',\alpha} = g_1^{f'(\alpha)}$ and shows that this is a commitment to $h(\alpha)$ where $h(X)$ is the polynomial committed in b.

The Prover computes $\beta := f(X) \pmod{(X-\alpha)}$ and sends the element $a_0 := g_1^{\beta}$ along with a proof (`ZKPoConst`) that this is a commitment to a known constant. He then sends the element $a_2 := g_1^{q(s)\cdot(s-\alpha)^2}$ along with a proof (`ZKPoKE`) that he knows the polynomial exponent between $g_1^{(s-\alpha)^2}$ and a_2. Lastly, he sends the verifiably sends the element $g_1^{f'(\alpha)}$.

The Verifier then verifies the equation

$$a = a_2 \cdot (g_1^{f'(\alpha)})^{s-\alpha} \cdot a_0$$

via the suitable pairing checks. This is a HVZK for the following relation:

$$\mathcal{R}_{\texttt{Der}}[g_1,\ (a,b)] = \{(a, b \in \mathbb{G}_1),\ f(X) \in \mathbb{F}_p[X]) : g_1^{f(s)} = a\ ,\ g_1^{f'(s)} = b\}.$$

Protocol 2.1. *Zero-knowledge proof of the derivative of a polynomial* (`ZKPoDer`)
Parameters: A pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$; generators $g_1,\ g_2$ for $\mathbb{G}_1,\ \mathbb{G}_2$ respectively.
Inputs: Elements $a, b \in \mathbb{G}_1$
Claim: The Prover knows a polynomial $f(X) \in \mathbb{F}_p[X]$ such that $g_1^{f(s)} = a$ and $g_1^{f'(s)} = b$.

1. The Fiat-Shamir heuristic generates a challenge α.
2. The Prover $\mathcal{P}$ computes the polynomial $q(X)$ and the constant $\beta \in \mathbb{F}_p$ such that

$$f(X) = q(X) \cdot (X - \alpha)^2 + f'(\alpha) \cdot (X - \alpha) + \beta.$$

3. $\mathcal{P}$ sends the elements
$$a_0 := g_1^{\beta} \ , \ C_{f'(\alpha)} := g_1^{f'(\alpha)}$$
along with (batchable) proofs for $\texttt{ZKPoConst}[g_1,\ a_0]$ and $\texttt{ZKPoConst}[g_1,\ C_{f'(\alpha)}]$.

4. $\mathcal{P}$ sends the element $a_2 := g_1^{q(s)\cdot(s-\alpha)^2}$ along with a proof for $\texttt{ZKPoKE}[g_1^{(s-\alpha)^2},\ a_2]$.

5. $\mathcal{P}$ sends a proof for $\texttt{ZKPoKE}[g_1^{s-\alpha},\ b \cdot C_{f'(\alpha)}^{-1}]$.

6. $\mathcal{V}$ verifies the ZKPoConsts, the ZKPoKEs and the equation
$$\mathbf{e}(C_{f'(\alpha)}\ ,\ g_2^{s-\alpha}) \stackrel{?}{=} \mathbf{e}(a \cdot a_2^{-1} \cdot a_0^{-1}\ ,\ g_2).$$ □

Lemma 2.2. *For a polynomial $f(X)$ and an element $\alpha \in \mathbb{F}_p$, the following are equivalent:*

1. $f(X) = c_0 \cdot (X-\alpha) \cdot f'(X)$ *for some* $c_0 \in \mathbb{F}_p^*$, $\alpha \in \mathbb{F}_p$
2. $f(X) = c \cdot (X-\alpha)^n$ *for some* $c \in \mathbb{F}_p^*$, $\alpha \in \mathbb{F}_p$, $n \in \mathbb{Z}^{\geq 0}$.

Proof. (1) $\Rightarrow$ (2): This is vacuous when $\deg(f(X)) = 1$. We now proceed by induction on the degree of $f(X)$. Let $n \geq 2$ be the degree of $f(X)$ and suppose the statement is true for polynomials of degree $\leq n-1$.

Differentiating both sides yields
$$f'(X) = c_0 \cdot [(X-\alpha) \cdot f''(X) + f'(X)]$$
and hence,
$$f'(X) = c_0 \cdot (1-c_0)^{-1} \cdot (X-\alpha) \cdot f''(X).$$
Since $\deg(f'(X)) < n$, it follows that $f'(X) = c_1 \cdot (X-\alpha)^{n-1}$ for some $c_1 \in \mathbb{F}_p^*$. This yields
$$f(X) = c_0 \cdot c_1 \cdot (X-\alpha)^{n-1},$$
which completes the proof.

(2) $\Rightarrow$ (1): This direction is trivial. □

The derivative $f'(X)$ has degree one less than that of $f(X)$. If $f(X)$ is divisible by $f'(X)$, the quotient $f(X) \cdot f'(X)^{-1}$ is a linear polynomial and hence, is of the form $c_0 \cdot (X-\alpha)$ for some $c \in \mathbb{F}_p^*$, $\alpha \in \mathbb{F}_p$. So $f(X) = c_0 \cdot (X-\alpha) \cdot f'(X)$ and hence, $f(X) = c \cdot (X-\alpha)^{\deg(f)}$ for some $c \in \mathbb{F}_p^*$.

For such a polynomial $f(X) = c \cdot (X-\alpha)^{\deg(f)}$, we have
$$f(0) = 0 \iff \alpha = 0 \iff f(X) = c \cdot X^{\deg(f)}.$$

Similarly,
$$f(0) = 0 \ \wedge \ f(1) = 1 \iff \alpha = 0 \ \wedge \ c = 1 \iff f(X) = X^{\deg(f)}.$$

Protocol 2.3. *Zero-knowledge proof of linear power* (ZKPoLinPow)
Parameters: A pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$; generators $g_1,\ g_2$ for $\mathbb{G}_1,\ \mathbb{G}_2$ respectively.
Inputs: Elements $a \in \mathbb{G}_1$
Claim: $a = g_1^{c\cdot(s-\alpha)^n}$ for some $c, \alpha \in \mathbb{F}_p$ and $n \in \mathbb{Z}$ known to the Prover.

1. The Prover $\mathcal{P}$ computes $a' = g_1^{c\cdot n\cdot(s-\alpha)^{n-1}}$ and sends it to the Verifier $\mathcal{V}$ along with a proof

of $\texttt{ZKPoDer}[g_1,\ (a, a')]$.

2. $\mathcal{P}$ generates a non-interactive proof of $\texttt{ZKPoKE}[a',\ a]$ and sends it to $\mathcal{V}$.

3. $\mathcal{V}$ accepts if and only if the ZKPoKE and the ZKPoDer are valid. □

To show that a committed polynomial $f(X)$ is $c \cdot X^n$ for *some* integer $n \geq 0$ and a constant c, a Prover needs to demonstrate the following properties:

1. $f(X)$ is divisible by the derivative $f'(X)$.
2. $f(0) = 0$.

For such a monomial $f(X) = c \cdot X^n$, the degree n is given by $n = f'(1) \cdot f(1)^{-1}$. In particular, if $f(X) = X^n$, the degree n is given by $n = f'(1)$.

This is a HVZK for the following relation:

$$\mathcal{R}_{\texttt{DegMono}}[g_1,\ (a, a_{\text{deg}})] = \{(a, a_{\text{deg}} \in \mathbb{G}_1),\ n \in \mathbb{Z}^{\geq 0},\ c \in \mathbb{F}_p) :\ g_1^{c \cdot s^n} = a\ ,\ g_1^n = a_{\text{deg}}\}$$

We will later use this as a building block for a more general protocol (ZKPoDeg) that allows a Prover to demonstrate that the degree of an arbitrary committed polynomial coincides with a committed integer $\leq$ length(CRS).

Protocol 2.4. *Zero-knowledge proof of degree of monomial* (ZKPoDegMono)
Parameters: A pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$; generators $g_1,\ g_2$ for $\mathbb{G}_1,\ \mathbb{G}_2$ respectively.
Inputs: Elements $a, a_{\text{deg}} \in \mathbb{G}_1$
Claim: $a = g_1^{c \cdot s^n}$, $a_{\text{deg}} = g_1^n$ for some $n \in \mathbb{Z}^{\geq 0}$, $c \in \mathbb{F}_p$ known to the Prover.

1. The Prover $\mathcal{P}$ sends the elements

$$a' := g_1^{c \cdot n \cdot s^{n-1}}\ \ ,\ \ a'_1 := (a')^s = g_1^{c \cdot n \cdot s^n}$$

along with a proof for $\texttt{ZKPoDer}[g_1,\ (a,\ a')]$.

2. $\mathcal{P}$ sends a zero-knowledge proof of equality of constant discrete logarithms (Chaum-Pedersen) between $[g_1,\ a_{\text{deg}}]$ and $[a,\ a'_1]$.

3. $\mathcal{V}$ accepts if and only if the ZKPoDer and the Chaum-Pedersen proofs are valid and the equation

$$\mathbf{e}(a'_1\ ,\ g_2) \stackrel{?}{=} \mathbf{e}(a'\ ,\ g_2^s)$$

holds. □

2.2 Linking a committed polynomial to its degree

The next protocol allows a Prover to succinctly demonstrate that a committed polynomial $f(X)$ is of degree less than a committed integer n. It hinges on the simple observation that the following are equivalent for any integer n and committed polynomial $f(X)$:

1. $\deg(f) < n$

2. The rational function $X^n \cdot f(X^{-1})$ is a polynomial divisible by X.

Given commitments $a = g_1^{f(s)}$, $b = g_1^n$, the Prover sends a commitment $\widetilde{g}_1 := g_1^{s^n}$ to X^n and uses the subprotocol `ZKPoDegMono` to show that this is a commitment to X^n, where n is the integer committed in b.

The Prover then sends a commitment $a^\vee := g_1^{h(s)}$ to the polynomial $h(X) := X^n \cdot f(X^{-1})$ along with a proof for $\texttt{ZKPoKE}[g_1^s,\ a^\vee]$. This demonstrates that $a^\vee$ is a commitment to a polynomial divisible by X.

For a randomly generated challenge γ, the Prover *verifiably* sends the element $g_1^{f(\gamma)}$. along with a proof that this is a commitment to the evaluation of $f(X)$ at γ. He also sends the element $a_\gamma := g_1^{s^n \cdot f(\gamma)}$ and uses the Chaum-Pedersen protocol to show that the discrete logarithm between the pair $(g_1^{s^n},\ a_\gamma)$ is a constant that coincides with the discrete logarithm between the pair $(g_1,\ g_1^{f(\gamma)})$.

The Prover then shows that the polynomial $h(X)$ committed in $a^\vee$ is such that

$$h(X) \equiv X^n \cdot f(\gamma) \pmod{(X - \gamma^{-1})}.$$

He does so by producing a $(s - \gamma^{-1})$-th root of $a^\vee \cdot a_\gamma^{-1}$. Since γ is randomly and uniformly generated, this implies that with overwhelming probability, $h(X) = X^n \cdot f(X^{-1})$.

This is a HVZK for the following relation:

$$\mathcal{R}_{\texttt{DegUp}}[g_1,\ (a_1, a_2)] = \left\{ \begin{array}{l} (a_1, a_2 \in \mathbb{G}_1),\ f(X) \in \mathbb{F}_p[X],\ n \in \mathbb{Z}) : \\ g_1^{f(s)} = a_1\ ,\ g_1^n = a_2\ ,\ \deg(f) < n \end{array} \right\}$$

Protocol 2.5. *Zero-knowledge proof of degree upper bound* (`ZKPoDegUp`)
Parameters: A pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$; generators $g_1,\ g_2$ for $\mathbb{G}_1,\ \mathbb{G}_2$ respectively.
Inputs: Elements $a, b \in \mathbb{G}_1$
Claim: The Prover knows an integer n and a polynomial $f(X)$ such that

$$a = g_1^{f(s)}\ \ ,\ \ b = g_1^n\ \ ,\ \ \deg(f) < n.$$

1. The Prover $\mathcal{P}$ chooses a random $c \in \mathbb{F}_p$ and sends $\widetilde{g}_1 := g_1^{c \cdot s^n}$ along with a proof of $\texttt{ZKPoDegMono}[g_1,\ (\widetilde{g}_1,\ b)]$.
2. $\mathcal{P}$ sends the elements
$$a^\vee := g_1^{c \cdot s^n \cdot f(s^{-1})}$$
along with a proof for $\texttt{ZKPoKE}[g_1^s,\ a^\vee]$
3. The Fiat-Shamir heuristic generates a challenge γ.
4. $\mathcal{P}$ sends the elements
$$C_{f,\gamma} := g_1^{f(\gamma)}\ \ ,\ \ a_\gamma := g_1^{c \cdot s^n \cdot f(\gamma)}$$
along with a zero-knowledge proof of equality of constant discrete logarithms (Chaum-Pedersen) between $(g_1,\ C_{f,\gamma})$ and $(\widetilde{g}_1,\ a_\gamma)$.
5. $\mathcal{P}$ sends proofs for $\texttt{ZKPoKE}[g_1^{s-\gamma},\ a \cdot C_{f,\gamma}^{-1}]$ and $\texttt{ZKPoKE}[g_1^{s-\gamma^{-1}},\ a^\vee \cdot a_\gamma^{-1}]$
6. $\mathcal{V}$ verifies the `ZKPoKE`s, the Chaum-Pedersen proof and the `ZKPoDegMono`. □

Protocol 2.6. *Zero-knowledge proof of degree upper bound* (`ZKPoDegUp`)
Parameters: A pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$; generators $g_1,\ g_2$ for $\mathbb{G}_1,\ \mathbb{G}_2$ respectively.
Inputs: Elements $a, b \in \mathbb{G}_1$
Claim: The Prover knows an integer n and a polynomial $f(X)$ such that

$$a = g_1^{f(s)} \ \ , \ \ b = g_1^{n} \ \ , \ \ \deg(f) < n.$$

1. The Prover $\mathcal{P}$ chooses a random $c \in \mathbb{F}_p$ and sends $\widetilde{g}_1 := g_1^{c \cdot s^n}$ along with a proof of `ZKPoDegMono`$[g_1,\ (\widetilde{g}_1,\ b)]$.
2. $\mathcal{P}$ sends the elements
$$a^{\vee} := g_1^{c \cdot s^n \cdot f(s^{-1})}$$
along with a proof for `ZKPoKE`$[g_1^s,\ a^{\vee}]$
3. The Fiat-Shamir heuristic generates a challenge γ.
4. $\mathcal{P}$ sends the elements
$$C_{f,\gamma} := g_1^{f(\gamma)} \ \ , \ \ a_{\gamma} := g_1^{c \cdot s^n \cdot f(\gamma)}$$
along with a zero-knowledge proof of equality of constant discrete logarithms (Chaum-Pedersen) between $(g_1,\ C_{f,\gamma})$ and $(\widetilde{g}_1,\ a_{\gamma})$.
5. $\mathcal{P}$ sends proofs for `ZKPoKE`$[g_1^{s-\gamma},\ a \cdot C_{f,\gamma}^{-1}]$ and `ZKPoKE`$[g_1^{s-\gamma^{-1}},\ a^{\vee} \cdot a_{\gamma}^{-1}]$
6. $\mathcal{V}$ verifies the `ZKPoKE`s, the Chaum-Pedersen proof and the `ZKPoDegMono`. □

We note that the proofs of bounded degrees are inherently batchable. If polynomials $f_1(X), \cdots, f_j(X)$ are all of degree less than n, then with overwhelming probability, the sum

$$f_{\gamma}(X) := \sum_{i=1}^{j} f_i(X) \cdot \gamma^i$$

is a polynomial of degree $< n$. Given committed polynomials $f_1(X), \cdots, f_j(X)$, a Prover can demonstrate this degree upper bound by showing that for a randomly generated challenge γ, the aggregated polynomial $f_{\gamma}(X)$ is such that the rational function $X^n \cdot f_{\gamma}(X^{-1})$ is a polynomial divisible by X.

We now move to the main protcol in this section. This protocol links a polynomial commitment to a commitment to its degree. It allows a Prover to prove in zero knowledge that given commitments to a polynomial $f(X)$ and an integer n, the committed polynomial is of degree n. More formally, given elements $a, a_{\deg} \in \mathbb{G}_1$, the Prover knows a polynomial $f(X)$ such that

$$a = g_1^{f(s)} \ \ , \ \ a_{\deg} = g_1^{\deg(f)}.$$

In particular, this can be used to show that two committed polynomials are of the same degree without revealing this common degree. More importantly, it enables the subsequent protocols in this paper.

The Prover starts out by isolating the leading term

$$\texttt{Coef}(f(X)\ ,\ \deg(f)) \cdot X^{\deg(f)}$$

of $f(X)$, providing a commitment to this monomial and using the protocol `ZKPoDegMono` to show that this is a commitment to a monomial of a degree that coincides with an integer committed in $a_{\deg}$. The Prover then sends a commitment to the polynomial

$$f(X) - \texttt{Coef}(f(X)\ ,\ \deg(f)) \cdot X^{\deg(f)}$$

obtained by removing the leading term and uses the protocol for the degree upper bound (`ZKPoDegUp`) to show that this is a commitment to a polynomial of a degree *strictly* less than the integer committed in $a_{\deg}$.

This is a HVZK for the following relation:

$$\mathcal{R}_{\texttt{Deg}}[g_1,\ (a, a_{\deg})] = \left\{ \left((a_1, a_2 \in \mathbb{G}_1),\ f(X) \in \mathbb{F}_p[X] \right) : g_1^{f(s)} = a\ ,\ g_1^{\deg(f)} = a_{\deg} \right\}$$

Protocol 2.7. *Zero-knowledge proof of degree* (`ZKPoDeg`)
Parameters: A pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$; generators $g_1,\ g_2$ for $\mathbb{G}_1,\ \mathbb{G}_2$ respectively.
Inputs: Elements $a, a_{\deg} \in \mathbb{G}_1$
Claim: The Prover knows an integer n and a polynomial $f(X)$ of degree n such that

$$a = g_1^{f(s)}\ \ ,\ \ a_{\deg} = g_1^{n}.$$

1. The Prover $\mathcal{P}$ isolates the leading term $c_n \cdot X^n$ of $f(X)$ and sends the elements

$$a_1 := g_1^{c_n \cdot s^n}\ \ ,\ \ a_2 := g_1^{f(s) - c_n \cdot s^n}\ \ \in\ \ \mathbb{G}_1.$$

2. $\mathcal{P}$ sends proofs for $\texttt{ZKPoDegMono}[g_1,\ (a_1,\ a_{\deg})]$ and $\texttt{ZKPoDegUp}[g_1,\ (a_2,\ a_{\deg})]$.
3. $\mathcal{V}$ verifies the `ZKPoDegUp`, the `ZKPoDegMono` and the equation $a \stackrel{?}{=} a_1 \cdot a_2$. □

2.3 The reverse polynomial

For a polynomial $f(X)$, we define the reverse polynomial

$$f^{\text{Rev}}(X) := X^{\deg(f)} \cdot f(X^{-1}) = \sum_{i=0}^{\deg(f)} \texttt{Coef}(f(X)\ ,\ i) \cdot X^{\deg(f)-i},$$

which is a polynomial of degree $\leq \deg(f)$. The degree of this reverse polynomial is given by

$$\deg(f^{\text{Rev}}(X)) = \deg(f) - \max\{m \in \mathbb{Z}\ :\ f(X) \equiv 0\ (\text{mod}\ X^m)\}.$$

We note that the map

$$\mathbb{F}_p[X] \longrightarrow \mathbb{F}_p[X]\ \ ,\ \ f(X) \mapsto f^{\text{Rev}}(X)$$

is not injective since for any integer k, $f(X) \cdot X^k$ has the same reverse as $f(X)$. But the map does yield an involution when restricted to the set of polynomials co-prime to X.

The next protocol allows a Prover to show that given two polynomial commitments, the second is a reverse of the first. This is an HVZK for the relation

$$\mathcal{R}_{\texttt{Rev}}[g_1,\ (a, b)] = \left\{ \left((a, b \in \mathbb{G}_1),\ f(X) \in \mathbb{F}_p[X] \right) : g_1^{f(s)} = a\ ,\ g_1^{s^{\deg(f)} \cdot f(s^{-1})} = b \right\}$$

We will need this protocol repeatedly for the subsequent protocols, including the Hadamard product and the range proof. It hinges on the observation that given two polyonomials $f(X)$ and $h(X)$, the following are equivalent with overwhelming probability:

1. $h(X) = f^{\mathrm{Rev}}(X)$

2. For a randomly generate challenge γ, the polynomial $h(X) - X^{\deg(f)} \cdot f(\gamma)$ is divisible by $X - \gamma^{-1}$.

To this end, the Prover *verifiably* sends a commitment to $X^{\deg(f)}$ using the protocols `ZKPoDeg` and `ZKPoDegMono`. In response to a challenge γ, he verifiably sends a commitment to the constant $f(\gamma)$ and the polynomial $f(\gamma) \cdot X^{\deg(f)}$. He then shows that $h(X) - X^{\deg(f)} \cdot f(\gamma)$ is divisible by $X - \gamma^{-1}$.

Protocol 2.8. *Zero-knowledge proof of reverse polynomial* (`ZKPoRev`)
Parameters: A pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$; generators $g_1,\ g_2$ for $\mathbb{G}_1,\ \mathbb{G}_2$ respectively.
Inputs: Elements $a, b \in \mathbb{G}_1$
Claim: The Prover knows a polynomial $f(X) \in \mathbb{F}_p[X]$ such that

$$a = g_1^{f(s)} \ \ , \ \ b = g_1^{s^{\deg(f)} \cdot f(s^{-1})}$$

1. The Prover $\mathcal{P}$ sends (batchable) proofs for `ZKPoKE`$[g_1,\ a]$ and `ZKPoKE`$[g_1,\ b]$.
2. $\mathcal{P}$ sends the elements
$$a_{\deg} := g_1^{\deg(f)} \ \ , \ \ \widetilde{g}_1 := g_1^{s^{\deg(f)}}$$
along with (batchable) proofs for `ZKPoDeg`$[g_1,\ (a,\ a_{\deg})]$ and `ZKPoDegMono`$[g_1,\ (\widetilde{g}_1,\ a_{\deg})]$.
3. The Fiat-Shamir heuristic generates a challenge γ.
4. $\mathcal{P}$ sends the elements
$$C_{f,\gamma} := g_1^{f(\gamma)} \ \ , \ \ Q_\gamma := g_1^{[f(s)-f(\gamma)]/[s-\gamma]} \ \in \ \mathbb{G}_1$$
5. $\mathcal{P}$ sends the element
$$b_\gamma := \widetilde{g}_1^{f(\gamma)} = g_1^{s^{\deg(f)} \cdot f(\gamma)} \ \in \ \mathbb{G}_1$$
along with a zero-knowledge proof of equality of constant discrete logarithms (Chaum-Pedersen) between $(g_1,\ C_{f,\gamma})$ and $(\widetilde{g}_1,\ b_\gamma)$.
6. $\mathcal{P}$ computes the polynomial
$$\widehat{q}_\gamma(X) := X^{\deg(f)} \cdot \frac{f(X^{-1}) - f(\gamma)}{X - \gamma^{-1}}$$
and sends the element $\widehat{Q}_\gamma := g_1^{\widehat{q}_\gamma(s)}$.
7. The Verifier $\mathcal{V}$ verifies the `ZKPoKE`s, the `ZKPoDeg`, the Chaum-Pedersen proof and the (batchable) equations
$$\mathbf{e}(Q_\gamma\ ,\ g_2^{s-\gamma}) \stackrel{?}{=} \mathbf{e}(a\ ,\ g_2) \ \ , \ \ \mathbf{e}(\widehat{Q}_\gamma\ ,\ g_2^{s-\gamma^{-1}}) \stackrel{?}{=} \mathbf{e}(b \cdot b_\gamma^{-1}\ ,\ g_2).$$

□

3 Coefficients at hidden positions

The next protocol allow a Prover to prove succinctly and in zero-knowledge that given elements $a, a_1, a_2 \in \mathbb{G}_1$, he knows a polynomial $f(X)$, an integer $n \leq \deg(f)$ and a field element α such

that X^n has coefficient α in $f(X)$ and a, a_1, a_2 are commitments to $f(X)$, n and α respectively. This is made possible by the protocols linking a committed polynomial to its degree.

The Prover decomposes the polynomial $f(X)$ as a sum

$$f(X) = f_-(X) + \alpha \cdot X^n + f_+(X) \cdot X^{n+1}$$

of three parts such that $\deg(f_-) < n$. In other words,

$$f_-(X) := f(X) \pmod{X^n} \ , \ f_+(X) := \frac{f(X) - f_-(X) - \alpha \cdot X^n}{X^{n+1}}.$$

The homomorphic property implies that the commitment to $f(X)$ is a product of commitments to $f_-(X)$, $\alpha \cdot X^n$ and $f_+(X) \cdot X^{n+1}$.

The first part (i.e. the low degree part) can be isolated using the protocol (`ZKPoDegUp`). The third part (i.e. the high degree part) can be isolated by showing that it is a multiple of X^{n+1}, where n is the integer committed in $a_{\texttt{Pos}}$. This leaves us with a commitment to the monomial $\alpha \cdot X^n$.

The Prover can use the protocol `ZKPoDegMono` that show that this is a commitment $a_{\texttt{mid}}$ to a *monomial* such that this monomial evaluates to α at $X = 1$, where α is the field element committed in $a_{\texttt{Coef}}$

This is a HVZK for the following relation:

$$\mathcal{R}_{\texttt{Coef}}[g_1, \ (a, \ a_{\texttt{Pos}}, \ a_{\texttt{Coef}})] = \left\{ \begin{array}{l} \big((a, \ a_{\texttt{Pos}}, \ a_{\texttt{Coef}} \ \in \ \mathbb{G}_1), \\ f(X), \ f_-(X), \ f_+(X) \in \mathbb{F}_p[X] \ , \ n \in \mathbb{Z}^{\geq 0} \ , \ \alpha \in \mathbb{F}_p\big) : \\ g_1^{f(s)} = a \ , \ g_1^n = a_{\texttt{Pos}} \ , \ g_1^\alpha = a_{\texttt{Coef}} \\ f(X) = f_-(X) + \alpha \cdot X^n + f_+(X) \\ \deg(f_-(X)) < n \ , \ f_+(X) \equiv 0 \pmod{X^{n+1}} \end{array} \right\}$$

Protocol 3.1. *Zero-knowledge proof of polynomial coefficient* (`ZKPoCoef`)
Parameters: A pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$; generators $g_1, \ g_2$ for $\mathbb{G}_1, \ \mathbb{G}_2$ respectively.
Inputs: Elements $a, a_{\texttt{Pos}}, a_{\texttt{Coef}} \in \mathbb{G}_1$
Claim: The Prover knows an integer n, a polynomial $f(X)$ of degree $\geq n$ and a field element α such that:

- $a = g_1^{f(s)}$, $a_{\texttt{Pos}} = g_1^n$, $a_{\texttt{Coef}} = g_1^\alpha$
- The coefficient of $f(X)$ at X^n is α.

1. $\mathcal{P}$ sends the elements

$$a_{\texttt{mid}} := g_1^{\alpha \cdot s^n} \ , \ a_{\texttt{mid},1} := a_{\texttt{mid}}^s = g_1^{\alpha \cdot s^{n+1}}$$

along with a proof for `ZKPoDegMono`$[g_1, \ (\widetilde{g_1}, \ a_{\texttt{Pos}})]$.
2. $\mathcal{P}$ sends a proof for `ZKPoKE`$[g_1^{s-1}, \ a_{\texttt{mid}} \cdot a_{\texttt{Pos}}^{-1}]$.
3. The Prover $\mathcal{P}$ computes the polynomials

$$f_-(X) := f(X) \pmod{X^n} \ , \ f_+(X) := f(X) - f_-(X) - \alpha \cdot X^n.$$

4. $\mathcal{P}$ sends the element $a_- := g_1^{f_-(s)}$ along with a proof for `ZKPoDegUp`$[g_1, \ (a_-, a_{\texttt{Pos}})]$.
5. $\mathcal{P}$ sends the element $a_+ := g_1^{f_+(s)}$ along with a proof for `ZKPoKE`$[a_{\texttt{mid},1}, \ a_+]$.

6. $\mathcal{V}$ verifies the `ZKPoDegUp`, the `ZKPoDegMono`, `ZKPoKE`, and the equations

$$a \stackrel{?}{=} a_- \cdot a_{\text{mid}} \cdot a_+ \ , \ \mathbf{e}(g_1, \ , \ a_{\texttt{mid},1}) \stackrel{?}{=} \mathbf{e}(g_1^s \ , \ a_{\texttt{mid}})$$

□

4 The Hadamard product

For vectors $v = (v_0, \cdots, v_n)$, $w = (w_0, \cdots, w_n)$ of length $n+1$, the *Hadamard product* (also known as the entrywise product) is defined as

$$v \odot w := (v_0 \cdot w_0, \cdots, v_n \cdot w_n) \ \in \ \mathbb{F}_p^{n+1}$$

If the lengths are unequal, we pad the shorter vector with zero entries on the right so that this product is well-defined.

For polynomials

$$f_1(X) = \sum_{i=0}^{\deg(f_1)} c_{i,1} X^i \ , \ f_2(X) = \sum_{i=0}^{\deg(f_2)} c_{i,2} X^i,$$

the *Hadamard product* $f_1 \odot f_2(X)$ (or $f_1(X) \odot f_2(X)$) is given by

$$f_1 \odot f_2(X) := \sum_{i=0}^{\min(\deg(f_1)\ ,\ \deg(f_2))} (c_{i,1} \cdot c_{i,2}) \cdot X^i.$$

Clearly, the Hadamard product is associative, commutative and bilinear. The dot product $f_1 \circ f_2$ is just the evaluation of the Hadamard product $f_1 \odot f_2(X)$ at $X = 1$, i.e.

$$f_1 \odot f_2(1) = f_1(X) \circ f_2(X).$$

Likewise, the dot product of two vectors is the sum of entries of the Hadamard product.

Let $\mathcal{I}$ be a index set and as before, let $\chi_{\mathcal{I}}(X)$ be the binary polynomial

$$\chi_{\mathcal{I}}(X) := \sum_{i \in \mathcal{I}} X^i.$$

Clearly,

$$f(X) \odot \chi_{\mathcal{I}}(X) = \sum_{i \in \mathcal{I}} \texttt{Coef}(f(X) \ , \ i) \cdot X^i.$$

The Hadamard product fulfills the following properties (among others):

1. $f(X) \odot \chi_{\mathcal{I}}(X) = 0 \iff \texttt{Coef}(f(X) \ , \ i) = 0 \ \forall \ i \in \mathcal{I}$

This will be useful for batched membership proofs.

2. $\exists \ f_{\mathcal{I}}^{\vee}(X) \in \mathbb{F}_p[X] \ : \ f(X) \odot f_{\mathcal{I}}^{\vee}(X) = \chi_{\mathcal{I}}(X) \iff \texttt{Coef}(f(X) \ , \ i) \neq 0 \ \forall \ i \in \mathcal{I}$

This will be useful for batched non-membership proofs.

3. $f(X) \odot \left[\frac{X^{\deg(f)+1}-1}{X-1} - f(X) \right] = 0 \iff f(X)$ is binary, i.e. $\texttt{Coef}(f(X) \ , \ i) \in \{0, 1\} \ \forall \ i$

This will be useful to construct a zero-knowledge range proof.

We now describe the observations underlying the protocol. As before, the reverse polynomial $f_2^{\text{Rev}}(X)$ is given by

$$f_2^{\mathrm{Rev}}(X) := X^{\deg(f_2)} \cdot f_2(X^{-1}) = \sum_{i=0}^{\deg(f_2)} \mathtt{Coef}(f_2(X)\ ,\ i) \cdot X^{\deg(f_2)-i}.$$

For a randomly generated challenge γ, the product

$$f_{\Pi,\gamma}(X)\ :=\ f_1(\gamma X) \cdot f_2^{\mathrm{Rev}}(X)$$

is a polynomial of degree

$$\deg(f_{\Pi,\gamma}) =\ \ \deg(f_1) + \deg(f_2) - \mathrm{val}_{(X)}(f_2(X))\ \ \leq\ \deg(f_1) + \deg(f_2).$$

Its coefficient $\mathtt{Coef}(f_{\Pi,\gamma}\ ,\ \deg(f_2))$ at $X^{\deg(f_2)}$ is given by the sum

$$\sum_{i=0}^{\min(\deg(f_1)\ ,\ \deg(f_2))} \mathtt{Coef}(f_1\ ,\ i) \cdot \mathtt{Coef}(f_2\ ,\ i) \cdot \gamma^i\ \ ,$$

which happens to coincide with the evaluation of the Hadamard product $f_1 \odot f_2(X)$ at γ. We exploit this simple fact in conjunction with the protocol for the coefficient at a hidden position (`ZKPoCoef`) to get a protocol for the Hadamard product.

Description of the protocol: The Prover starts out by *verifiably* sending commitments to the reverse polynomial $f_2^{\mathrm{Rev}}(X)$ of $f_2(X)$ and to the degree of $f_2(X)$. He uses the protocols `ZKPoRev` and `ZKPoDeg` to do so.

In response to a challenge γ, he verifiably sends a commitment $C_{f_{1,2}(\gamma)}$ to the evaluation of $f_{1,2}$ at γ. He *verifiably* sends a commitment to the product

$$f_{\Pi,\gamma}(X) := f_1(\gamma X) \cdot f_2^{\mathrm{Rev}}(X)$$

by combining the protocols `ZKPoTwist` and `ZKPoProd`.

The Prover now uses the protocol `ZKPoCoef` to show that this polynomial $f_{\Pi,\gamma}(X)$ has a coefficient at the position $X^{\deg(f_2)}$ that coincides with the constant committed in $C_{f_{1,2}(\gamma)}$.

Thus, to summarize, the Prover combines the protocols `ZKPoCoef`, `ZKPoProd`, `ZKPoDeg`, `ZKPoConst`, `ZKPoRev`, `ZKPoTwist` to succinctly prove that $f_{1,2}(X) = f_1 \odot f_2(X)$. This is a HVZK for the relation

$$\mathcal{R}_{\mathtt{HadProd}}[g_1,\ (a_1, a_2),\ a_{1,2}] = \left\{ \begin{array}{l} \big((a_1,\ a_2,\ a_{1,2} \in \mathbb{G}_1)\ ,\ f_1(X),\ f_2(X) \in\ \mathbb{F}_p[X]\big) : \\ g_1^{f_1(s)} = a_1\ ,\ g_1^{f_2(s)} = a_2\ ,\ g_1^{f_1 \odot f_2(s)} = a_{1,2} \end{array} \right\}$$

Protocol 4.1. *Zero-knowledge proof of the Hadamard Product* (`ZKPoHadProd`)
Parameters: A pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$; generators $g_1,\ g_2$ for $\mathbb{G}_1,\ \mathbb{G}_2$ respectively.
Inputs: Elements $a_1, a_2, a_{1,2}\ \in\ \mathbb{G}_1$
Claim: The Prover knows polynomials $f_1(X),\ f_2(X)$ such that:

$$g_1^{f_1(s)} = a_1\ \ ,\ \ g_1^{f_2(s)} = a_2\ \ ,\ \ g_1^{f_1 \odot f_2(s)} = a_{1,2}$$

($f_1 \odot f_2$ denotes the Hadamard product)

1. $\mathcal{P}$ sends

$$a_2^{\vee} := g_1^{f_2^{\mathrm{Rev}}(s)}\ \ ,\ \ a_{\deg,2} := g_1^{\deg(f_2)}$$

along with proofs for $\mathtt{ZKPoDeg}[g_1,\ (a_2,\ a_{\deg,2})]$, $\mathtt{ZKPoRev}[g_1,\ (a_2,\ a_2^{\vee})]$.

2. The Fiat-Shamir heuristic generates a challenge γ.

3. $\mathcal{P}$ sends the element $C_{f_{1,2}(\gamma)} := g_1^{f_1 \odot f_2(\gamma)}$ along with proofs for $\texttt{ZKPoConst}[g_1,\ C_{f_{1,2}(\gamma)}]$ and $\texttt{ZKPoKE}[g_1^{s-\gamma},\ a_{1,2} \cdot C_{f_{1,2}(\gamma)}^{-1}]$.

4. $\mathcal{P}$ sends $a_{1,\gamma} := g_1^{f_1(\gamma \cdot s)}$ along with a proof for $\texttt{ZKPoTwist}[g_1,\ (a_1, \gamma),\ a_{1,\gamma}]$.

5. $\mathcal{P}$ computes the polynomial

$$f_{\Pi,\gamma}(X) \ := \ f_1(\gamma \cdot X) \ \cdot \ f_2^{\text{Rev}}(X)$$

and sends the element

$$a_{\Pi,\gamma} := g_1^{f_{\Pi,\gamma}(s)}$$

along with a proof for $\texttt{ZKPoProd}[g_1,\ (a_{1,\gamma},\ a^{\vee}),\ a_{\Pi,\gamma}]$.

6. $\mathcal{P}$ sends a proof for $\texttt{ZKPoCoef}[g_1,\ (a_{\Pi,\gamma}\ ,\ a_{\text{deg},2}\ ,\ C_{f_{1,2}(\gamma)})]$

7. $\mathcal{V}$ verifies the ZKPoCoef, the ZKPoProd, the ZKPoDegs, the ZKPoConst, the ZKPoRev, the ZKPoTwist and accepts if and only if they are all valid. □

We note that the Hadamard product protocol is *partially* batchable in the following sense. For indices $i = 1, \cdots, k$, the relations $\mathcal{R}_{\texttt{HadProd}}[g_1,\ (a_1,\ a_{i,2}),\ a_{i,1,2}]$ can be proven by demonstrating the single relation

$$\mathcal{R}_{\texttt{HadProd}}\big[g_1,\ (a_1,\ \prod_{i=1}^{k} a_{i,2}^{\gamma^i}),\ \prod_{i=1}^{k} a_{i,1,2}^{\gamma^i}\big]$$

for a challenge γ randomly generated by the Fiat-Shamir heuristic. This is because the following are equivalent with overwhelming probability:

1. $f_1(X) \ \odot \ f_{i,2}(X) \ = \ f_{i,1,2}(X)$ for each index $i = 1, \cdots, k$.
2. $f_1(X) \ \odot \ \big[\sum_{i=1}^{k} \gamma^i \cdot f_{i,2}(X)\big] \ = \ \big[\sum_{i=1}^{k} \gamma^i \cdot f_{i,1,2}(X)\big]$

Furthermore, if polynomials $f_{i,1}(X)$, $f_{i,2}(X)$ are such that

$$k \cdot (\deg(f_i, 1) + 1),\ k \cdot (\deg(f_i, 1) + 1) \ < \ \big|\texttt{CRS}\big|,$$

the relations

$$f_{i,1}(X) \ \odot \ f_{i,2}(X) \ = \ f_{i,1,2}(X)$$

can be proven by demonstrating the single relation

$$\big[\sum_{i=0}^{k-1} f_{i,1}(X) \cdot X^{N \cdot i}\big] \ \odot \ \big[\sum_{i=0}^{k-1} f_{i,2}(X) \cdot X^{N \cdot i}\big] \ = \ \big[\sum_{i=0}^{k-1} f_{i,1,2}(X) \cdot X^{N \cdot i}\big]$$

where $N := \max\{\deg(f_{i,1}),\ \deg(f_{i,2})\} + 1$.

4.1 Proof of bit coefficients

The next protocol allows a Prover to prove in zero-knowledge that a committed polynomial is *binary*, i.e. its coefficients all lie in $\{0, 1\}$. This, in turn, immediately yields a range proof. It will also be useful for a subsequent protocol that links a committed polynomial to a commitment

to the number of non-zero coefficients in this polynomial, which is akin to the size of committed data in a Vector Commitment.

A polynomial $f(X) = \sum_{i=0}^{n} c_i X^i$ is binary if and only if $c_1 \cdot (1 - c_i) = 0$ for each index i. This, in turn, is equivalent to the Hadamard product of $f(X)$ and

$$f^{\vee}(X) := \Big[\sum_{i=0}^{\deg(f)} X^i \Big] - f(X) = \Big[\frac{X^{\deg(f)+1} - 1}{X - 1} \Big] - f(X)$$

being zero.

The Prover sends a commitment $a^{\vee} := g_1^{f^{\vee}(s)}$ to this polynomial $f^{\vee}(X)$ and uses the protocols `ZKPoDeg` and `ZKPoDegMono` to show that $a \cdot a^{\vee}$ is a commitment to $\frac{X^{\deg(f)+1}-1}{X-1}$. He then sends a proof that the polynomials committed in a and $a^{\vee}$ have Hadamard product zero.

This is a HVZK for the following relation:

$$\mathcal{R}_{\texttt{Binary}}[g_1,\ a] = \left\{ \begin{array}{l} \big((a \in \mathbb{G}_1),\ f(X) \in \mathbb{F}_p[X]\big) : \\ g_1^{f(s)} = a \\ \texttt{Coef}(f(X)\ ,\ i) \in \{0,1\}\ \ \forall\ i \in [\deg(f)] \end{array} \right\}$$

Protocol 4.2. *Zero-knowledge proof of binary polynomial* (`ZKPoBinary`)
Parameters: A pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$; generators $g_1,\ g_2$ for $\mathbb{G}_1,\ \mathbb{G}_2$ respectively.
Inputs: Elements $a \in \mathbb{G}_1$
Claim: The Prover knows a polynomial $f(X)$ such that:
- all coefficients of $f(X)$ lie in $\{0, 1\}$
- $g_1^{f(s)} = a$

1. The Prover $\mathcal{P}$ sends the elements

$$a_{\deg} := g_1^{\deg(f)}\ \ ,\ \ \widetilde{g}_1 := g_1^{s^{\deg(f)}}$$

along with (batchable) proofs for $\texttt{ZKPoDeg}[g_1,\ (a,\ a_{\deg})]$, $\texttt{ZKPoDegMono}[g_1,\ (\widetilde{g}_1,\ a_{\deg})]$

2. $\mathcal{P}$ computes the polynomial

$$f^{\vee}(X) := \Big[\sum_{i=0}^{\deg(f)} X^i \Big] - f(X) = \Big[\frac{X^{\deg(f)+1} - 1}{X - 1} \Big] - f(X)$$

and sends the element $a^{\vee} := g_1^{f^{\vee}(s)}$.

3. $\mathcal{P}$ sends a proof for $\texttt{ZKPoHadProd}[g_1,\ (a,\ a^{\vee}),\ 1]$.

4. The Verifier $\mathcal{V}$ verifies the `ZKPoDeg`, the `ZKPoDegMono`, the `ZKPoHadProd` and the equation

$$\mathbf{e}(a \cdot a^{\vee}\ ,\ g_2^{s-1}) \stackrel{?}{=} \mathbf{e}(\widetilde{g}_1 \cdot g_1^{s-1}\ ,\ g_2)$$

□

Alternatively, the Prover can show that $f(X)$ and the polynomial

$$f^{\vee}(X) := \frac{X^{|\texttt{CRS}|} - 1}{X - 1} - f(X)$$

have Hadamard product zero. This reduces the effort involved in sending the `ZKPoDeg` proofs. We could have the Verifier store the element $\widetilde{g}_1 := g_1^{s^{|\texttt{CRS}|}}$ or have the Prover send this element along

with a succinct (non-ZK) proof of exponentiation. The Prover can pre-compute the commitment to $\frac{X^{|\mathtt{CRS}|}-1}{X-1}$ so that the commitment

$$a^{\vee} := g_1^{f^{\vee}(s)} = g_1^{[s^{|\mathtt{CRS}|}-1]/[s-1]} \cdot g_1^{-f(s)}$$

can be computed with just one scalar multiplication.

4.2 Batched membership proofs

Let

$$\mathbf{v} = (v_0, \cdots, v_n) \ , \ \mathbf{w} = (w_0, \cdots, w_m)$$

be vectors with entries in $\mathbb{F}_p$. As before, we commit to these vector by committing to the polynomials $\mathbf{v}(X)$, $\mathbf{w}(X)$ with coefficients v_i, w_i:

$$\mathtt{Com}(v) = g_1^{\sum_{i=0}^{n} v_i X^i} \quad , \quad \mathtt{Com}(w) = g_1^{\sum_{i=0}^{m} w_i X^i} \ .$$

As before, we commit to any index set $\mathcal{I}$ by committing to the binary polynomial $\sum_{i\in\mathcal{I}} X^i$. For such an index set $\mathcal{I}$, we say $\mathbf{v}$ and $\mathbf{w}$ agree at $\mathcal{I}$ if for every index $i \in \mathcal{I}$, we have $v_i = w_i$. Clearly,

$$v_i = w_i \ \forall \ i \ \in \ \mathcal{I} \iff v(X) \odot \chi_{\mathcal{I}}(X) = w(X) \odot \chi_{\mathcal{I}}(X).$$

Thus, a Prover can use the protocol `ZKPoHadProd` to prove - succinctly and in zero-knowledge - that $\mathbf{v}$ and $\mathbf{w}$ agree at every index $i \in \mathcal{I}$.

4.3 Disjointness of vectors and batched non-membership proofs

For two committed vectors $\mathbf{v} = (v_0, \cdots, v_n)$, $\mathbf{w} = (w_0, \cdots, w_n)$ and a set $\mathcal{I}$ of indices, we would like a protocol that allows a Prover to show that for each index $i \in \mathcal{I}$, $\mathbf{v}$ and $\mathbf{w}$ disagree at the position i, i.e. $v_i \neq w_i$. Since the quotient of the commitments to $\mathbf{v}$, $\mathbf{w}$ is a commitment to the difference $v - w$, this boils down to the task of showing that for every $i \in \mathcal{I}$, the i-th entry of $v - w$ is non-zero. Phrased in terms of polynomial commitments, the protocol should succinctly demonstrate that the coefficient of a committed polynomial at X^i is non-zero for every $i \in \mathcal{I}$.

Rather than such a protocol with the set $\mathcal{I}$ revealed in the clear, we would prefer a protocol where the Verifier sees a *commitment* to $\mathcal{I}$ rather than $\mathcal{I}$. This allows for better succinctness and zero-knowledge properties. The commitment that seems suitable for this purpose is the commitment

$$\mathtt{Com}(\mathcal{I}) = \mathtt{Com}(\chi_{\mathcal{I}}(X)) = g_1^{\sum_{i\in\mathcal{I}} s^i} \ ,$$

which is the [KZG10] commitment to the polynomial $\chi_{\mathcal{I}}(X) := \sum_{i\in\mathcal{I}} X^i$.

Thus, given elements $a, \ a_{\mathtt{nz}} \ \in \ \mathbb{G}_1$, the protocol should allow a Prover to show that he knows a polynomial $f(X)$ and a set $\mathcal{I}$ of indices such that:

- $g_1^{f(s)} = a$
- $g_1^{\chi_{\mathcal{I}}(s)} = a_{\mathtt{nz}}$, where $\chi_{\mathcal{I}}(X) := \sum_{i\in\mathcal{I}} X^i$.
- For every $i \in \mathcal{I}$, the coefficient of $f(X)$ at X^i is non-zero.

To this end, the Prover uses the protocol `ZKPoDeg` that the degree of the polynomial $f(X)$ coincides with the integer committed in a_{deg}. The Prover uses the protocol `ZKPoBinary` to

show that the polynomial committed in $\mathtt{C}_{\mathcal{I}}$ is a binary polynomials. This proves that $\mathtt{C}_{\mathcal{I}}$ is a commitment to $\chi_{\mathcal{I}}(X)$ for some set $\mathcal{I} \subseteq [\deg(f)]$

The protocol `ZKPoHadProd` is used to demonstrate that the polynomials $f_{\overline{\mathcal{I}}}(X)$ and $\chi_{\mathcal{I}}(X)$ have Hadamard product zero. In other words, for every index $i \in \mathcal{I}$, $f_{\overline{\mathcal{I}}}(X)$ has coefficient 0 at X^i. Thus, for every $i \in \mathcal{I}$, $f(X)$ and $f_{\mathcal{I}}(X)$ have the same coefficient at X^i.

An element of $\mathbb{F}_p$ being non-zero is equivalent to this element having a multiplicative inverse. Thus, the Prover computes the polynomial

$$f^{\vee}_{\mathcal{I}}(X) := \sum_{i \in \mathcal{I}} \mathtt{Coef}(f(X)\ ,\ i)^{-1} \cdot X^i$$

and sends a commitment $a^{\vee}_{\mathcal{I}} := g_1^{f^{\vee}_{\mathcal{I}}(s)}$ to this polynomial.

The Prover then uses the protocol `ZKPoHadProd` to show that the polynomials committed in $a_{\mathcal{I}}$ and $a^{\vee}_{\mathcal{I}}$ have Hadamard product $\chi_{\mathcal{I}}(X)$. Thus, if $a^{\vee}_{\mathcal{I}}$ is a commitment to the polynomial $\sum_{i=0} d_i X^i$, the coefficients d_i and c_i are such that $c_i \cdot d_i = 1$ for every index $i \in \mathcal{I}$. This, in turn, implies that c_i is non-zero for each $i \in \mathcal{I}$.

This is a HVZK for the relation:

$$\mathcal{R}_{\mathtt{NZCoef}}[g_1,\ a,\ a_{\mathtt{nz}}] = \left\{ \begin{array}{l} \big((a,\ a_{\mathtt{nz}}\ \in\ \mathbb{G}_1), \\ f(X) \in \mathbb{F}_p[X]\ ,\ \mathcal{I} \subseteq [\deg(f)]\big) : \\ g_1^{f(s)} = a\ ,\ g_1^{\chi_{\mathcal{I}}(s)} = a_{\mathtt{nz}}, \\ \forall\, i \in \mathcal{I},\ \mathtt{Coef}(f(X)\ ,\ i) \neq 0 \end{array} \right\}$$

$\mathtt{Coef}(f(X)\ ,\ i)$ denotes the coefficient of $f(X)$ at X^i. $\chi_{\mathcal{I}}(X)$ is the polynomial $\sum\limits_{i \in \mathcal{I}} X^i$.

Protocol 4.3. *Zero-knowledge proof of polynomial with non-zero coefficients at a index set* (`ZKPoNZCoef`)

Parameters: A pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$; generators $g_1,\ g_2$ for $\mathbb{G}_1,\ \mathbb{G}_2$ respectively.

Inputs: Elements $a,\ a_{\mathtt{nz}}\ \in\ \mathbb{G}_1$.

Claim: The Prover knows a polynomial $f(X)$ and a set $\mathcal{I}$ of indices such that:

- $g_1^{f(s)} = a$.
- $g_1^{\chi_{\mathcal{I}}(s)} = a_{\mathtt{nz}}$, where $\chi_{\mathcal{I}}(X) := \sum\limits_{i \in \mathcal{I}} X^i$.
- For every $i \in \mathcal{I}$, the coefficient of $f(X)$ at X^i is non-zero.

1. $\mathcal{P}$ sends a proof for $\mathtt{ZKPoBinary}[g_1,\ a_{\mathtt{nz}}]$.
2. $\mathcal{P}$ computes the polynomial

$$f^{\vee}_{\mathcal{I}}(X) := \sum_{i \in \mathcal{I}} \mathtt{Coef}(f(X)\ ,\ i)^{-1} \cdot X^i$$

and sends the element $a^{\vee}_{\mathcal{I}} := g_1^{f^{\vee}_{\mathcal{I}}(s)}$ along with a proof for $\mathtt{ZKPoHadProd}[g_1,\ (a,\ a^{\vee}_{\mathcal{I}}),\ a_{\mathtt{nz}}]$.

3. $\mathcal{V}$ verifies the the `ZKPoBinary` and the `ZKPoHadProd`. □

We say two vectors $\mathbf{v} = (v_0, \cdots, v_n) \in \mathbb{F}_p^{n+1}$, $\mathbf{w} = (w_0, \cdots, w_m) \in \mathbb{F}_p^{m+1}$ are *disjoint* if the coordinates at every position are unequal to each other, i.e. for every index i, $v_i \neq w_i$. If the

vectors are of different lengths, we pad the shorter vector by adding zeros to the right so that this notion is well-defined.

For our vector commitment that identifies the vectors $\mathbf{v}$, $\mathbf{w}$ with the polynomials $\sum v_i X^i$, $\sum w_i X^i$ respectively, this entails the coefficients v_i, w_i of X^i being unequal for every i.

The protocol discussed in this subsection allows a Prover to prove in zero-knowledge that two committed polynomials do not have the same coefficient at *any* position. Our primary applications are:

- batched non-membership proofs for key-value pairs with respect to a committed vector

- a protocol for the number of non-zero entries in a Vector Commitment

This is a special case of the preceding protocol with $\mathcal{I}$ as the entire set $[\deg(f)]$ rather than an arbitrary subset.

4.4 The subset-sum problem

We use the Hadamard product protocol to provide a succinct zero-knowledge protocol for the so-called *subset-sum problem*. Given a committed polynomial $f(X)$ and a committed value $T \in \mathbb{F}_p$, the Prover shows that he knows an index set $\mathcal{I}$ such that

$$\sum_{i \in \mathcal{I}} \texttt{Coef}(f(X)\ ,\ i) = T.$$

The Prover sends a commitment to the binary polynomial $\chi_{\mathcal{I}}(X) := \sum_{i \in \mathcal{I}} X^i$ and uses the protocol `ZKPoBinary` to show that this is a binary polynomial, thus demonstrating that this is a commitment to the characteristic polynomial of some index set. The Prover then sends a commitment to the Hadamard product

$$f_{\mathcal{I}}(X) := f(X)\ \odot\ \chi_{\mathcal{I}}(X)$$

and uses the protocol `ZKPoHadProd` to show that this is the Hadamard product of $f(X)$ and $\chi_{\mathcal{I}}(X)$. Lastly, the Prover shows - in zero-knowledge - that $f_{\mathcal{I}}(1) = T$.

The sum is independent of the sequence in which the coefficients appear in the polynomial and hence, this protocol works for this purpose. The analogous protocol for a *weighted* sum (aka knapsack problem) is more subtle. It hinges on a permutation argument, i.e. a protocol that demonstrates that given commitments to two polynomials $f_1(X)$, $f_2(X)$, the second is obtained from the first by permuting the coefficients of the first with a permutation known to the Prover.

5 Permutations of committed vectors

We now discuss a protocol that allows a Prover to show that for a multiset $\mathcal{M}$ committed as in the [Ngu05] bilinear accumulator

$$\texttt{Com}(g_1,\ \mathcal{M}) := g_1^{\prod\limits_{m \in \mathcal{M}} (s+m)^{\text{mult}(m,\mathcal{M})}}$$

and a polynomial $f(X)$ committed with the [KZG10] commitment

$$\texttt{Com}(g_1,\ f(X)) := g_1^{f(s)}\ \ ,$$

the multiset of coefficients of $f(X)$ coincides with $\mathcal{M}$. Note that this is equivalent to the equation

$$\prod_{i=0}^{\deg(f)} X + \texttt{Coef}(f(X)\ ,\ i)\ =\ \prod_{m\in\mathcal{M}} (X+m)^{\text{mult}(m,\mathcal{M})}.$$

By the Schwartz-Zippel lemma, this is equivalent to the equation

$$\prod_{i=0}^{\deg(f)} \gamma + \texttt{Coef}(f(X)\ ,\ i)\ =\ \prod_{m\in\mathcal{M}} (\gamma+m)^{\text{mult}(m,\mathcal{M})}$$

being valid for some randomly generated challenge γ.

We now describe the protocol. Write $f(X) = \sum_{i=0}^{\deg(f)} c_i X^i$. In response to a challenge $\gamma \in \mathbb{F}_p$ generated by the Fiat-Shamir heuristic, the Prover verifiably sends a commitment to the polynomial

$$f_\gamma(X) := \sum_{i=0}^{\deg(f)+1} (c_i+\gamma)\cdot X^i = f(X) + \gamma\cdot\frac{X^{\deg(f)+1}-1}{X-1}.$$

He then constructs and sends a commitment to a polynomial $\widetilde{f}(X)$ that fulfills the following property:

$$X\cdot\left[f_\gamma(X)\ \odot\ \widetilde{f}(X)\right] = \widetilde{f}(X) - 1$$

and subsequently proves that this equation holds using the Hadamard product protocol.

We claim that if the polynomial $\widetilde{f}(X)$ satisfies this property, the coefficient of $\widetilde{f}(X)$ at the position $X^{\deg(f)+1}$, coincides with the product

$$\prod_{i=0}^{\deg(f)} (c_i+\gamma).$$

We first explain why this is the case and then describe the construction of the polynomial $\widetilde{f}(X)$.

If the polynomial

$$\widetilde{f}(X) = \sum_{i=0}^{\deg(f)+1} \widetilde{c}_i X^i$$

satisfies the foregoing property, we have

$$\widetilde{c}_0 = 1\ \ ,\ \ \widetilde{c}_i = \widetilde{c}_{i-1}\cdot(c_i+\gamma)\ \ \forall\ i = 1,\cdots,\deg(f)+1.$$

Thus, by induction, we have

$$\widetilde{c}_i = \prod_{j=0}^{i-1}(c_i+\gamma)\ \ \forall\ i = 1,\cdots,\deg(f)+1,$$

as claimed. The Prover constructs $\widetilde{f}(X)$ as follows:

- Set $\widetilde{c}_0 = 1$
- For $i = 1,\cdots,\deg(f)+1$, set

$$\widetilde{c}_i = \widetilde{c}_{i-1}\cdot(c_{i-1}+\gamma) = \prod_{j=0}^{i-1}(c_j+\gamma).$$

The Prover then sends the commitment $g_1^{\widetilde{f}(s)}$ to the polynomial $\widetilde{f}(X)$. He sends a proof for the Hadamard product in addition to proofs for the following:

1. $\widetilde{f}(0) = 1$

2. $\widetilde{f}(X)$ is a polynomial of degree $1 + \deg(f)$ and its leading coefficient (i.e. the coefficient at $X^{\deg(\widetilde{f})+1}$) coincides with the value at γ of the polynomial committed in $\texttt{Com}(\mathcal{M})$.

This is an HVZK for the following relation:

$$\mathcal{R}_{\texttt{CoefMulSet}}[g_1,\ (a,\ A)] = \left\{ \begin{array}{l} \big((a,\ A\ \in\ \mathbb{G}_1), \\ n \in \big[\big|\texttt{CRS}\big|\big]\ ,\ \ c_0, \cdots, c_n \in \mathbb{F}_p\big) : \\ a = g_1^{\sum_{i=0}^{n} c_i \cdot s^i}\ \ ,\ \ A = g_1^{\prod_{i=0}^{n} (s+c_i)} \end{array} \right\}$$

Protocol 5.1. *Protocol for the coefficient multiset* (`ZKPoCoefMul`)
Parameters: A pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$; generators $g_1,\ g_2$ for $\mathbb{G}_1,\ \mathbb{G}_2$ respectively.
Inputs: Elements $a, A \in \mathbb{G}_1$

Claim: The Prover knows a polynomial $f(X) = \sum_{i=0}^{\deg(f)} c_i X^i$ such that

$$a = g_1^{f(s)}\ \ ,\ \ A = g_1^{\prod_{i=0}^{\deg(f)} (s+c_i)}$$

1. The Prover $\mathcal{P}$ sends the elements $\widetilde{g}_1 := g_1^{s^{\deg(f)}}$, $a_{\deg} := g_1^{\deg(f)}$ along with proofs for $\texttt{ZKPoDeg}[g_1,\ (a,\ a_{\deg})]$, $\texttt{ZKPoDegMono}[g_1,\ (\widetilde{g}_1,\ a_{\deg})]$
2. The Fiat-Shamir heuristic generates a challenge γ.
3. $\mathcal{P}$ computes the polynomial

$$f_\gamma(X) := f(X) + \gamma \cdot \frac{X^{|\texttt{CRS}|} - 1}{X - 1}$$

and sends the element $a_\gamma := g_1^{f_\gamma(s)}$.

4. $\mathcal{P}$ computes the polynomial

$$\widetilde{f}(X) := \sum_{i=0}^{\deg(f)+1} \prod_{j=0}^{i-1} (c_i + \gamma) \cdot X^i$$

and sends the elements

$$\widetilde{a} := g_1^{\widetilde{f}(s)}\ \ ,\ \ \widehat{a} := g_1^{[\widetilde{f}(s)-1]/[s-1]}.$$

5. $\mathcal{P}$ sends a proof for $\texttt{ZKPoKE}[g_1^s,\ \widetilde{a} \cdot g_1^{-1}]$
6. $\mathcal{P}$ sends a proof for $\texttt{ZKPoHadProd}[g_1,\ (a_\gamma,\ \widetilde{a}),\ \widehat{a}]$
7. $\mathcal{P}$ sends a proof for $\texttt{ZKPoCoef}[g_1,\ (\widetilde{a},\ a_{\deg} \cdot g_1,\ A)]$.
8. The Verifier $\mathcal{V}$ verifies the `ZKPoKE`, `ZKPoCoef`, the `ZKPoDegs`, the `ZKPoHadProd` and the (batchable) equations

$$\mathbf{e}(\widehat{a}\ ,\ g_2^{s-1}) \stackrel{?}{=} \mathbf{e}(\widetilde{a} \cdot g_1^{-1}\ ,\ g_2)\ \ ,\ \ \mathbf{e}(\widetilde{g}_1 \cdot g_1^{s-1}\ ,\ g_2) \stackrel{?}{=} \mathbf{e}(a^{-1} \cdot a_\gamma\ ,\ g_2^{s-1}).$$

□

We discuss two immediate implications of the protocol.

5.0.1 Permuted polynomials

For commitments

$$a = g_1^{f(s)} \ , \ b = g_1^{h(s)}$$

to two polynomials $f(X)$, $h(X)$, a Prover can succinctly demonstrate that $h(X)$ can be obtained by permuting the coefficients of $f(X)$. This hypothesis is equivelent to the two coefficient multisets coinciding, which is equivalent to the equation

$$\prod_{i=0}^{\deg(f)} X + \texttt{Coef}(f(X)\ ,\ i) \ = \ \prod_{i=0}^{\deg(h)} X + \texttt{Coef}(h(X)\ ,\ i).$$

The Prover sends the element

$$A := g_1^{\prod_{i=0}^{\deg(f)} (s+\texttt{Coef}(f(X)\ ,\ i))}.$$

He can then use the protocol `ZKPoCoefMul` for pairs (a, A) and (b, A) to show that the multiset formed by the coefficients of $f(X)$ coincides with the multiset committed in A. This implies, in particular, that the coefficients of $f(X)$ and $h(X)$ yield the same multiset.

We refer to this protocol as `ZKPoPerm`. Thus, $\texttt{ZKPoPerm}[g_1,\ (a, b)]$ is merely the conjunction of the protocols $\texttt{ZKPoCoefMul}[g_1,\ (a,\ A)]$ and $\texttt{ZKPoCoefMul}[g_1,\ (b,\ A)]$.

This is a HVZK for the following relation:

$$\mathcal{R}_{\texttt{Perm}}[g_1,\ (a,\ b)] = \left\{ \begin{array}{l} \big((a,\ b\ \in\ \mathbb{G}_1)\ ,\ \ f(X)\ \in\ \mathbb{F}_p[X] \\ \text{Permutation } \sigma :\ \big[|\texttt{CRS}|\big]\ \longrightarrow\ \big[|\texttt{CRS}|\big]\big) : \\ a = g_1^{f(s)}\ \ ,\ \ b = g_1^{f^\sigma(s)} \end{array} \right\}$$

5.0.2 Commitments to permutations

As before, for a permutation $\sigma :\ [0,\ n] \longrightarrow [0,\ n]$, we commit to σ by committing to the polynomial $S_\sigma(X) := \sum_{i=0}^{n} \sigma(i) \cdot X^i$. In particular, we commit to the identity permutation of $[0,\ n]$ by committing to the polynomial

$$P_{\texttt{id},n}(X) := \sum_{i=0}^{n} i \cdot X^i.$$

The polynomial $P_{\texttt{id},n}(X)$ satisfies the equation

$$P_{\texttt{id},n}(X) = X \cdot \frac{d}{dX}\ \Big[\frac{X^n - 1}{X - 1}\Big].$$

Hence, a Prover can combine the protocols `ZKPoDeg` and `ZKPoDer` to show that a $\mathbb{G}_1$-element is a commitment to a polynomial $P_{\texttt{id},n}(X)$ for *some* integer n.

We refer to this protocol as the proof of identity permutation or `ZKPoIdPerm` for short.

5.0.3 Non-repetition in a committed vector

The protocol also allows a Prover to show that a committed vector has no repetitions, i.e. no element occurs with multiplicity higher than one.

Consider a vector $\mathbf{v} = (v_0, \cdots, v_n)$ committed, as before, by treating the entries as polynomial coefficients and committing to the polynomial $\mathbf{v}(X) := \sum_{i=0}^{n} v_i X^i$:

$$a := \texttt{Com}(\mathbf{v}) := \texttt{Com}(g_1, \sum_{i=0}^{n} v_i X^i) = g_1^{\sum_{i=0}^{n} v_i s^i}.$$

The Prover can demonstrate non-repetition in $\mathbf{v}$ as follows:

1. Send a commitment $A \in \mathbb{G}_1$ to the polynomial $\prod_{i=0}^{n}(X + v_i)$.

2. Use the protocol `ZKPoCoefMulSet` to show that A is a commitment to the same multiset as the one formed by the coefficients of the polynomial $\sum_{i=0}^{n} v_i X^i$.

3. Send a commitment A' to the derivative $\mathbf{v}'(X)$ of $\mathbf{v}(X)$ and use the protocol `ZKDer` to show that this is a commitment to the derivative of the polynomial committed in A.

4. Show that the polynomials committed in A and A' are relatively prime.

5.0.4 The radical of a polynomial and the underlying set of a multiset

For a polynomial $f(X) \in \mathbb{F}_p[X]$, we define its *radical* $f_{\texttt{rad}}(X)$ as the product of all distinct irreducible factors of $f(X)$. Equivalently, $f_{\texttt{rad}}(X)$ is the unique monic polynomial that generates the (principal) ideal

$$\sqrt{\big(f(X)\big)} := \big\{h(X) \in \mathbb{F}_p[X] \ : \ h(X)^N \in \big(f(X)\big) \text{ for some } N \in \mathbb{Z}\big\} \ \subseteq \ \mathbb{F}_p[X].$$

If $f(X)$ is monic, the two polynomials are linked by the equation

$$f_{\texttt{rad}}(X) = \frac{f(X)}{\gcd(f(X)\ ,\ f'(X))}$$

where $f'(X)$ denotes the derivative. Thus, if a Verifier has access to two polynomial commitments, a Prover can combine the protocols `PoDer`, `PoGCD` and `PoProd` to succinctly show that the second commitment represents the radical of the polynomial represented by the first. We refer to this as the protocol `PoRad`.

In particular, consider the case of a multiset $\mathcal{M}$ be a multiset and its underlying set $\texttt{Set}(\mathcal{M})$ committed as in the [Ngu05] accumulator. The polynomials

$$f_{\mathcal{M}}(X) := \prod_{m \in \mathcal{M}} (X + m)^{\text{mult}(m,\, \mathcal{M})} \ \ , \ \ f_{\texttt{Set}(\mathcal{M})}(X) := \prod_{m \in \texttt{Set}(\mathcal{M})} (X + m).$$

are such that $f_{\texttt{Set}(\mathcal{M})}$ is the radical of $f_{\mathcal{M}}(X)$.

Thus, if a Verifier has access to two [Ngu05] multiset commitments, a Prover can combine the protocols `ZKPoDer`, `ZKPoGCD` and `ZKPoProd` to succinctly show that the second [Ngu05] commitment represents the underlying set of the multiset represented by the first.

5.0.5 Splitting of committed polynomials

The protocol `ZKPoCoefMulSet` also allows a Prover to show succinctly that a [KZG10] committed polynomial splits into linear factors in $\mathbb{F}_p[X]$. Let $h(X)$ be a committed polynomial that splits completely in $\mathbb{F}_p[X]$ and whose decomposition is known to the Prover. Multiplying $h(X)$ by a suitable constant if necessary, we may assume without loss of generality that $h(X)$ is monic.

Let

$$h(X) = \prod_{i=0}^{n}(X + c_i)$$

be the decomposition of $h(X)$. The Prover constructs the polynomial

$$f(X) := \sum_{i=0}^{n} c_i X^i$$

and sends a commitment to $f(X)$. He then uses the protocol `ZKPoCoefMulSet` to show that the coefficients of $f(X)$ yield a multiset that coincides with the multiset committed in $\texttt{Com}(g_1,\ h(X))$.

5.1 Permutation commitments

For a permutation $\sigma \in S_{n+1}$ of the set $\{0, 1, \cdots, n\}$, we define the *permutation polynomial*

$$S_\sigma(X) := \sum_{i=0}^{n} \sigma(i) \cdot X^i$$

and consider the element $a_\sigma := g_1^{S_\sigma(s)}$ to be a commitment to σ.

Note that this is (by design) different from the permutation commitment in the original PLONK, which commits to the polynomial valued $\omega^{\sigma(i)}$ at ω^i for a fixed n-th root of unity ω in $\mathbb{F}_p$. Our scheme, by contrast, does not require $\mathbb{F}_p^*$ to have a large 2-adic subgroup.

Given an element $g_1^{S_\sigma(s)}$, a Prover can show that this is a commitment to *some* permutation by showing that the committed polynomials has the same coefficients as the set $\{0, \cdots, n\}$. Using the protocol described in the preceding subsection, this would entail showing that the coefficients of $S_\sigma(X)$ coincide with the set of elements committed in the element

$$g_1^{\prod_{i=0}^{n}(s+i)}$$

via the [Ngu05] multiset accumulator.

The right-shift: For a polynomial $P(X) = \sum_{i=0}^{\deg(P)} e_i \cdot X^i$, we call the polynomial

$$\texttt{RShift}(P)(X) := e_{\deg(P)} + \sum_{i=0}^{\deg(P)-1} e_i X^{i+1}$$

the *cyclic right shift* of $P(X)$. The right shift can be defined as

$$\texttt{RShift}(P)(X) := X \cdot P(X) \pmod{X^{\deg(P)+1} - 1}.$$

We now describe a protocol to show that the polynomials $f(X)$, $\widetilde{f}(X)$ are such that $\widetilde{f}(X) = f^\sigma(X)$ in the sense that

$$\texttt{Coef}(\widetilde{f}(X)\ ,\ i) = \texttt{Coef}(f(X)\ ,\ \sigma(i))\ \ \forall\, i$$

We assume the Verifier has access to the commitments

$$a := g_1^{f(s)}\ ,\ \widetilde{a} := g_1^{\widetilde{f}(s)}\ ,\ a_\sigma := g_1^{S_\sigma(s)}.$$

Write

$$f(X) = \sum_{i=0}^{n} c_i X^i \quad , \quad \widetilde{f}(X) = \sum_{i=0}^{n} \widetilde{c}_i X^i.$$

In response to two challenges β, γ generated by the Fiat-Shamir heuristic, the Prover verifiably sends a commitment to the polynomial

$$\begin{aligned} f_{\beta,\gamma}(X) &:= f(X) + \beta \cdot \big(\sum_{i=0}^{n} i \cdot X^i\big) + \gamma \cdot \sum_{i=0}^{n} X^i \\ &= \sum_{i=0}^{n} \big(c_i + \beta \cdot i + \gamma\big) \cdot X^i. \end{aligned}$$

Note that $X \cdot \sum_{i=0}^{n} i \cdot X^i$ is the derivative of the polynomial $\frac{X^{n+3}-1}{X-1}$, which makes it easy to *verifiably* send a commitment to $\sum_{i=0}^{n} i \cdot X^i$ using the protocol `ZKPoDer` in case this is not part of the preprocessed setup.

Similarly, the Prover verifiably sends a commitment to the polynomial

$$\begin{aligned} \widetilde{f}_{\beta,\gamma}(X) &:= \widetilde{f}(X) + \beta \cdot S_\sigma(X) + \gamma \cdot \sum_{i=0}^{n} X^i \\ &= \sum_{i=0}^{n} \big(\widetilde{c}_i + \beta \cdot \sigma(i) + \gamma\big) \cdot X^i. \end{aligned}$$

It is straightforward for the Prover to show that the commitments to $f_{\beta,\gamma}(X)$ and $\widetilde{f}_{\beta,\gamma}(X)$ are well-formed.

The Prover now constructs a polynomial $P(X)$ such that $P(X)$ and the right shift of $P(X)$ satisfy the equation

$$f_{\beta,\gamma}(X) \ \odot \ P(X) \ = \ \widetilde{f}_{\beta,\gamma}(X) \ \odot \ \texttt{RShift}(P(X)),$$

where $\odot$, as before, denotes the Hadamard product. We first describe this construction and then explain why it allows the Prover to demonstrate that $f(X)$ and $\widetilde{f}(X)$ are linked by the permutation σ.

The Prover constructs $P(X) = \sum_{i=0}^{n} e_i \cdot X^i$ as follows:

- $e_0 = 1$.
- For every $i \in [n-1]$, $e_{i+1} = e_i \cdot \frac{[c_i + \beta \cdot i + \gamma]}{[\widetilde{c}_i + \beta \cdot \sigma(i) + \gamma]}$.

Note that, by construction,

$$\begin{aligned} e_n \cdot \frac{[c_n + \beta \cdot n + \gamma]}{[\widetilde{c}_n + \beta \cdot \sigma(n) + \gamma]} \ &= \ \prod_{i=0}^{n} \frac{[c_i + \beta \cdot i + \gamma]}{[\widetilde{c}_i + \beta \cdot \sigma(i) + \gamma]} \\ &= \ 1 \ = \ e_0. \end{aligned}$$

The Prover sends a commitment to $P(X)$ and to its right-shift

$$\texttt{RShift}(P(X)) := X \cdot P(X) \ (\text{mod } X^{n+1} - 1)$$

along with a proof that the second polynomial is a right-shift of the first. He sends a zero-knowledge proof that $P(0) = 1$, which is equivalent to the constant term of $P(X)$ being 1.

The Prover now uses the Hadamard product protocol (`ZKPoHadProd`) to show that

$$f_{\beta,\gamma}(X) \odot P(X) = \widetilde{f}_{\beta,\gamma}(X) \odot \texttt{RShift}(P(X)).$$

The last equation implies that

$$\prod_{i=0}^{n}(c_i + \beta \cdot i + \gamma) = \prod_{i=0}^{n}(\widetilde{c}_i + \beta \cdot \sigma(i) + \gamma).$$

Since the challenges β, γ were randomly and uniformly generated, the following lemma implies that with overwhelming probability, $\widetilde{c}_i = c_{\sigma(i)}$ for each i.

Lemma 5.2. *Let $c_0, \cdots, c_n$, $\widetilde{c}_0, \cdots, \widetilde{c}_n$ be elements of $\mathbb{F}_p$. Let σ be a permutation of $[n]$. For random elements $\beta, \gamma \in \mathbb{F}_p$, if*

$$\prod_{i=0}^{n}(c_i + \beta \cdot i + \gamma) = \prod_{i=0}^{n}(\widetilde{c}_i + \beta \cdot \sigma(i) + \gamma) \ ,$$

then with overwhelming probability, $\widetilde{c}_i = c_{\sigma(i)}$ for every i.

A proof of this lemma can be found in the appendix of [GWC19]. It hinges on the Schwartz-Zippel lemma.

5.2 The knapsack problem

We discuss a generalization of the aforementioned subset-sum problem, which is the so-called *weighted subset-sum* or the *knapsack* problem. For a committed polynomial $f(X)$, a committed polynomial $w(X)$ and a committed total value $T \in \mathbb{F}_p$, this protocol allows a Prover to show that he knows an index set

$$\mathcal{I} = \{n_0, \cdots, n_{\deg(w)}\} \subseteq [\deg(f)]$$

such that

$$\sum_{i=0}^{\deg(w)} \texttt{Coef}(f(X) \ , \ n_i) \cdot \texttt{Coef}(w(X) \ , \ i) = T.$$

The Prover chooses a permutation $\sigma : [\deg(f)] \longrightarrow [\deg(f)]$ such that $\sigma(i) = n_i$ for every $i \in [\deg(w)]$. He sends a commitment to the polynomial

$$f^{\sigma}(X) := \sum_{i=0}^{n} \texttt{Coef}(f(X) \ , \ \rho(i)) \cdot X^i$$

and uses the protocol `ZKPoPerm` to show that this polynomial is a permutation of $f(X)$.

The Prover sends a commitment to the Hadamard product $f^{\sigma}(X) \odot w(X)$ and uses the protocol `ZKHadPoProd` to show that this is the Hadamard product of $f^{\sigma}(X)$ and $w(X)$. Lastly, the Prover shows that this Hadamard product evaluates to T at $X = 1$.

6 Common subsequences at hidden index sets in committed vectors

We discuss a protocol that allows a Prover to succinctly show that given two $\mathbb{F}_p$-vectors committed via our vector commitment scheme and two committed index sets, the subsequences cut out by these index sets coincide. In other words, for committed polynomials

$$f(X) = \sum_{i=0}^{\deg(f)} c_i X^i \ , \ \widetilde{f}(X) = \sum_{i=0}^{\deg(\widetilde{f})} \widetilde{c}_i X^i$$

and (hidden) committed index sets $\mathcal{I}$, $\widetilde{\mathcal{I}}$, the protocol allows a Prover to show that the sequences

$$(c_i)_{i\in\mathcal{I}} \ , \ (\widetilde{c}_i)_{i\in\widetilde{\mathcal{I}}}$$

coincide. As a special case, this yields a protocol to show that for two committed vectors, one occurs as a subsequence of the other.

This is an HVZK for the following relation:

$$\mathcal{R}_{\texttt{Subseq}}[g_1,\ (a,\ \widetilde{a}),\ (C_{\mathcal{I}}.\ C_{\widetilde{\mathcal{I}}})] = \left\{ \begin{array}{l} \big((a,\ \widetilde{a},\ C_{\mathcal{I}},\ C_{\widetilde{\mathcal{I}}} \ \in \ \mathbb{G}_1), \\ f(X),\ \widetilde{f}(X) \ \in \ \mathbb{F}_p[X] \ , \ \mathcal{I},\ \widetilde{\mathcal{I}} \ \subseteq \ [|\texttt{CRS}| - 1] \\ \text{Permutation } \sigma :\ [\max(\mathcal{I}\cup\widetilde{\mathcal{I}})] \ \longrightarrow [\max(\mathcal{I}\cup\widetilde{\mathcal{I}})]\big) : \\ a = g_1^{f(s)} \ , \ b = g_1^{\widetilde{f}(s)} \ , \ C_{\mathcal{I}} = g_1^{\chi_{\mathcal{I}}(s)} \ , \ C_{\widetilde{\mathcal{I}}} = g_1^{\chi_{\widetilde{\mathcal{I}}}(s)} \\ [f(X)\odot\chi_{\mathcal{I}}(X)]^{\sigma} \ = \ [\widetilde{f}(X)\odot\chi_{\widetilde{\mathcal{I}}}(X)] \\ \forall\ i,j\in\mathcal{I}, \quad i<j \iff \sigma(i)<\sigma(j) \end{array} \right\}$$

Note that the Hadamard products

$$f_{\mathcal{I}}(X) := f(X) \ \odot \ \chi_{\mathcal{I}}(X) \ , \ \widetilde{f}_{\widetilde{\mathcal{I}}}(X) := \widetilde{f}(X) \ \odot \ \chi_{\widetilde{\mathcal{I}}}(X)$$

have the same non-zero coefficients and in the same order. Thus, there exists an efficiently computable permutation σ of $[\max(\mathcal{I}\cup\widetilde{\mathcal{I}})]$ such that

$$f_{\mathcal{I}}^{\sigma}(X) = \widetilde{f}_{\widetilde{\mathcal{I}}}(X).$$

The Prover can use the permutation protocol `ZKPoPerm` to show that he knows such a permutation. However, this would not be enough to convince the Verifier that this permutation σ does not change the order in which the elements c_i ($i \in \mathcal{I}$) occur. For this, we will need the notion of a permutation that is *increasing on an index set*, which we now define.

Definition 6.1. *For an index set $\mathcal{I} \subseteq [N]$ and a permutation τ of $[N]$, we say τ is increasing on $\mathcal{I}$ if for any distinct $i, j \in \mathcal{I}$ the following holds:*

$$i < j \iff \tau(i) < \tau(j)$$

The Prover succinctly demonstrates the following:

1. The Hadamard products $f_{\mathcal{I}}(X) := f(X) \ \odot \ \chi_{\mathcal{I}}(X) \ , \ \widetilde{f}_{\widetilde{\mathcal{I}}}(X) := \widetilde{f}(X) \ \odot \ \chi_{\widetilde{\mathcal{I}}}(X)$ are linked by a committed permutation $\sigma :\ [\max(\mathcal{I}\cup\widetilde{\mathcal{I}})] \longrightarrow [\max(\mathcal{I}\cup\widetilde{\mathcal{I}})]$.

2. This committed permutation σ is increasing on the index set $\mathcal{I}$.

The first can be done using the protocol `ZKPoPerm`. We discuss the second part in the next subsection.

6.1 Commitments to Increasing Permutations

As before, for a permutation $\sigma :\ [n] \longrightarrow [n]$, we commit to σ by committing to the polynomial $S_\sigma(X) := \sum_{i=0}^{n} \sigma(i) \cdot X^i$. We now describe a protocol that allows a Prover to show that a committed permutation σ is increasing on a (hidden) committed index set $\mathcal{I}$. We assume the Verifier has access to commitments to $S_\sigma(X)$ and $\chi_{\mathcal{I}}(X)$.

The Prover uses the protocol `ZKPoPerm` to show that the committed polynomial $S_\sigma(X)$ represents a permutation of $[n+1]$. He then uses the Hadamard product protocol to *verifiably* send a commitment to the polynomial

$$S_{\sigma,\mathcal{I}}(X) := S_\sigma(X) \odot \chi_{\mathcal{I}}(X) = \sum_{i\in\mathcal{I}} \sigma(i)\cdot X^i.$$

The multiset of coefficients of $S_{\sigma,\mathcal{I}}(X)$ is the set $\sigma(\mathcal{I}) \subseteq [n+1]$. The Prover constructs a polynomial $\widetilde{S}_{\sigma,\mathcal{I}}(X)$ as follows:

- for every $i \in \mathcal{I}$, set

$$\texttt{Coef}(\widetilde{S}_{\sigma,\mathcal{I}}(X)\ ,\ i) \ :=\ \texttt{Coef}(S_{\sigma,\mathcal{I}}(X)\ ,\ i) \ =\ \sigma(i).$$

- for every i in the complement $\big[\max(\mathcal{I})\big] \setminus \mathcal{I}$, set $m_i := \max\{j \in \mathcal{I} : j < i\}$ and set

$$\texttt{Coef}(\widetilde{S}_{\sigma,\mathcal{I}}(X)\ ,\ i) \ :=\ \texttt{Coef}(S_{\sigma,\mathcal{I}}(X)\ ,\ m_i) \ :=\ \sigma(m_i).$$

Thus, by construction, the coefficients of $S_{\sigma,\mathcal{I}}(X)$ all lie in $\sigma(\mathcal{I})$ and are in *non-decreasing* order. The Prover sends a commitment to this polynomial $\widetilde{S}_{\sigma,\mathcal{I}}(X)$ and uses the Hadamard product protocol to show that

$$\widetilde{S}_{\sigma,\mathcal{I}}(X) \odot \chi_{\mathcal{I}}(X) \ =\ S_{\sigma,\mathcal{I}}(X).$$

He then shows that the coefficients of $\widetilde{S}_{\sigma,\mathcal{I}}(X)$ are in non-decreasing order. This will imply that, in particular, the coefficients of $S_{\sigma,\mathcal{I}}(X)$ are in non-decreasing order and since the coefficients of $S_{\sigma,\mathcal{I}}(X)$ are known to be distinct, it will then follow that these coefficients are, in fact, in increasing order.

We now describe how the Prover demonstrates the non-decreasing nature of the coefficients of $\widetilde{S}_{\sigma,\mathcal{I}}(X)$. The Prover *verifiably* sends a commitment to the product

$$(1-X)\cdot\widetilde{S}_{\sigma,\mathcal{I}}(X) \ =\ -\texttt{Coef}(S_{\sigma,\mathcal{I}}\ ,\ 0) + \sum_{i=0}^{\max(\mathcal{I})} \big[\texttt{Coef}(S_{\sigma,\mathcal{I}}\ ,\ i+1) - \texttt{Coef}(S_{\sigma,\mathcal{I}}\ ,\ i)\big]\cdot X^{i+1}.$$

The Prover then uses the protocol for coefficient multisets (`PoCoefMulSet`) and the protocol for the radical (ZKPoRad) to show that this committed polynomial $(1-X)\cdot S_{\sigma,\mathcal{I}}(X)$ is such that its coefficients form a multiset whose underlying set is contained in the interval $[n+1]$.

This is a HVZK for the following relation:

$$\mathcal{R}_{\texttt{Perm}\uparrow}[g_1,\ (a,\ b),\ C_{\mathcal{I}}] = \left\{ \begin{array}{l} \big((a,\ b,\ C_{\mathcal{I}}\ \in\ \mathbb{G}_1), \\ f(X)\ \in\ \mathbb{F}_p[X]\ \ ,\ \ \mathcal{I} \subseteq [|\texttt{CRS}|-1] \\ \text{Permutation } \sigma:\ \big[|\texttt{CRS}|\big]\ \longrightarrow\ \big[|\texttt{CRS}|\big]\big): \\ a = g_1^{f(s)}\ \ ,\ \ b = g_1^{f^\sigma(s)}\ \ ,\ \ C_{\mathcal{I}} = g_1^{\chi_{\mathcal{I}}(s)} \\ \forall\ i,j \in \mathcal{I},\ i<j\ \iff\ \sigma(i)<\sigma(j) \end{array} \right\}$$

7 A variant of PLONK via coefficients

We discuss a variant of Plonk via coefficients instead of Lagrange bases. The two primary ingredients are:

1. The Hadamard product protocol which allows a Prover to show that the multiplication gates are properly aligned.

2. A permutation protocol that shows that two committed polynomials are linked by a (publicly known) permutation. This is necessary to account for the copy constraints in the Plonk proof system.

Arithmetic circuits: The *Plonk arithmetization* for a fan-in 2 circuit is given by the equation

$$\mathbf{Q}_{L,i} \cdot c_{L,i} + \mathbf{Q}_{R,i} \cdot c_{R,i} + \mathbf{Q}_{O,i} \cdot c_{O,i} + \mathbf{Q}_{\mathcal{M},i} \cdot c_{L,i} \cdot c_{R,i} + \mathbf{Q}_{\text{const}} = 0$$

where the vectors $\mathbf{Q}_L$, $\mathbf{Q}_R$, $\mathbf{Q}_O$, $\mathbf{Q}_{\mathcal{M}}$, $\mathbf{Q}_{\text{const}}$ are the public *selector vectors*.

The selector vectors are committed as polynomials. We diverge from the original Plonk scheme by treating the vector entries as coefficients instead of polynomial values at roots of unity. This is consistent with our design goal of avoiding the need for a large smooth- order subgroup in the scalar field. Thus, we construct the arithmetic circuit as follows:

1. Let $c_{L,i}$, $c_{R,i}$, $c_{O,i}$ denote the left/right/output wire of the i-th gate respectively. We define polynomials

$$f_L(X) := \sum c_{L,i} \cdot X^i \ , \ f_R(X) := \sum c_{R,i} \cdot X^i \ , \ f_O(X) := \sum c_{O,i} \cdot X^i.$$

2. If the i-th gate is a multiplication gate, set

$$\texttt{Coef}(\mathbf{Q}_L(X) \ , \ i) = 0 \ , \ \texttt{Coef}(\mathbf{Q}_R(X) \ , \ i) = 0 \ , \ \texttt{Coef}(\mathbf{Q}_{\mathcal{M}}(X) \ , \ i) = -1 \ , \ \texttt{Coef}(\mathbf{Q}_O(X) \ , \ i) = 1$$

3. If the i-th gate is an addition gate, set

$$\texttt{Coef}(\mathbf{Q}_L(X) \ , \ i) = 1 \ , \ \texttt{Coef}(\mathbf{Q}_R(X) \ , \ i) = 1 \texttt{Coef}(\mathbf{Q}_{\mathcal{M}}(X) \ , \ i) = 0 \ , \ \texttt{Coef}(\mathbf{Q}_O(X) \ , \ i) = -1.$$

Thus, the equation is given by

$$\mathbf{Q}_L(X) \odot f_L(X) + \mathbf{Q}_R(X) \odot f_R(X) + \mathbf{Q}_O(X) \odot f_O(X) + \mathbf{Q}_{\mathcal{M}}(X) \odot f_L(X) \odot f_R(X) + \mathbf{Q}_{\text{const}} = 0,$$

where $\odot$ denotes the Hadamard (entrywise) product.

Addition gates: Let $\mathcal{A}$ be the index set corresponding to the multiplication gates and let $\chi_{\mathcal{A}}(X)$ be its characteristic polynomial. The relation for the multiplication gates can be translated to:

$$[f_L(X) + f_R(X)] \odot \chi_{\mathcal{A}}(X) = f_O(X) \odot \chi_{\mathcal{A}}(X).$$

Multiplication gates: Let $\mathcal{M}$ be the index set corresponding to the multiplication gates and let $\chi_{\mathcal{M}}(X)$ be its characteristic polynomial. The relation for the multiplication gates can be translated to:

$$[f_L(X) \odot f_R(X)] \odot \chi_{\mathcal{M}}(X) = f_O(X) \odot \chi_{\mathcal{M}}(X).$$

Boolean gates: Let $\mathcal{B}$ be the index set corresponding to the Boolean gates and let $\chi_{\mathcal{B}}(X)$ be its characteristic polynomial. The relation for the Boolean constraints can be translated to:

$$f_L(X) \odot \Big[\frac{X^{\max(\mathcal{B})+1} - 1}{X - 1} - f_L(X)\Big] \odot \chi_{\mathcal{B}}(X) = 0.$$

Inversion gates: Let $\mathcal{I}_{\text{inv}}$ be the index set corresponding to the inversion gates and let $\chi_{\mathcal{I}_{\text{inv}}}(X)$ be its characteristic polynomial. The relation for the inversion constraints can be translated to:

$$f_L(X) \odot f_O(X) \odot \chi_{\mathcal{I}_{\text{inv}}}(X) = \chi_{\mathcal{I}_{\text{inv}}}(X).$$

Public inputs: Let $\texttt{Pub}$ be the index set corresponding to the public inputs and let $\chi_{\texttt{Pub}}(X)$ be its characteristic polynomial. Let c_i ($i \in \texttt{Pub}$) be the public inputs. The relation for the public constraints can be translated to:

$$f_L(X) \odot \chi_{\texttt{Pub}}(X) = \sum_{i \in \texttt{Pub}} c_i X^i.$$

7.1 A permutation argument for copy constraints

We first discuss a permutation argument necessary for the copy constraints in a PLONK-type arithmetization. Let $f(X)$, $\widetilde{f}(X)$ be polynomials of degree n in $\mathbb{F}_p[X]$. Let $\sigma \in S_{n+1}$ be a publicly known permutation. Set

$$S_\sigma(X) := \sum_{i=0}^{n} \sigma(i) \cdot X^i.$$

We call this the *permutation polynomial for the circuit* and consider the $\mathbb{G}_1$-element

$$a_\sigma := g_1^{S_\sigma(s)}$$

to be a commitment to σ. Note that this is (by design) different from the permutation commitment in the original PLONK, which commits to the polynomial valued $\sigma(i)$ at ω^i for a fixed n-th root of unity ω in $\mathbb{F}_p$. Our scheme, by contrast, does not require $\mathbb{F}_p^*$ to have a large 2-adic subgroup. In fact, to avoid vulnerability to the Cheon-type attacks, we prefer that $\mathbb{F}_p^*$ does not have large smooth-order subgroups.

The right shift: For a polynomial $P(X) = \sum_{i=0}^{\deg(P)} e_i \cdot X^i$, we call the polynomial

$$\texttt{RShift}(P)(X) := e_{\deg(P)} + \sum_{i=0}^{n-1} e_i X^{i+1}$$

the *cyclic right shift* of $P(X)$. The right shift can be defined as

$$\texttt{RShift}(P)(X) := X \cdot P(X) \pmod{X^{\deg(P)+1} - 1}$$

and can be computed without any $\mathbb{F}_p$-arithmetic. It satisfies the equation

$$\texttt{RShift}(P)(X) - X \cdot P(X) = e_{\deg(P)} \cdot (X^{\deg(P)+1} - 1).$$

We now describe a protocol to show that the polynomials $f(X)$, $\widetilde{f}(X)$ are such that $\widetilde{f}(X) = \sigma(f)(X)$ in the sense that

$$\texttt{Coef}(\widetilde{f}(X)\ ,\ i) = \texttt{Coef}(f(X)\ ,\ \sigma(i))\ \ \forall\ i.$$

We assume the Verifier has access to the commitments

$$a := g_1^{f(s)}\ ,\ \widetilde{a} := g_1^{\widetilde{f}(s)}\ ,\ a_\sigma := g_1^{S_\sigma(s)}.$$

Write

$$f(X) = \sum_{i=0}^{n} c_i X^i\ \ ,\ \ \widetilde{f}(X) = \sum_{i=0}^{n} \widetilde{c}_i X^i$$

where $n = \max(\deg(f),\ \deg(\widetilde{f}))$. Note that we do not assume the c_i and the $\widetilde{c}_i$ are non-zero and hence, the polynomials $f(X)$, $\widetilde{f}(X)$ might have distinct degrees.

In response to two challenges β, γ generated by the Fiat-Shamir heuristic, the Prover verifiably sends a commitment to the polynomial

$$f_{\beta,\gamma}(X)\ :=\ f(X) + \beta \cdot \big(\sum_{i=0}^{n} i \cdot X^i\big) + \gamma \cdot \sum_{i=0}^{n} X^i\ =\ \sum_{i=0}^{n} \big(c_i + \beta \cdot i + \gamma\big) \cdot X^i.$$

Note that $X \cdot \sum_{i=0}^{n} i \cdot X^i$ is the derivative of the polynomial $\frac{X^{n+3}-1}{X-1}$, which makes it easy to *verifiably* send a commitment to $\sum_{i=0}^{n} i \cdot X^i$ using the protocol `ZKPoDer` in case this is not part of the preprocessed setup.

Similarly, the Prover verifiably sends a commitment to the polynomial

$$\widetilde{f}_{\beta,\gamma}(X) \;:=\; \widetilde{f}(X) + \beta \cdot S_\sigma(X) + \gamma \cdot \sum_{i=0}^{n} X^i \;=\; \sum_{i=0}^{n} \left(\widetilde{c}_i + \beta \cdot \sigma(i) + \gamma\right) \cdot X^i.$$

It is straightforward for the Prover to show that the commitments to $f_{\beta,\gamma}(X)$ and $\widetilde{f}_{\beta,\gamma}(X)$ are well-formed.

The Prover now constructs a polynomial $P(X)$ with constant term 1 and such that $P(X)$ and its cyclic right shift satisfy the equation

$$f_{\beta,\gamma}(X) \;\odot\; P(X) \;=\; \widetilde{f}_{\beta,\gamma}(X) \;\odot\; \texttt{RShift}(P(X)),$$

where $\odot$, as before, denotes the Hadamard product. We first describe this construction and then explain why it allows the Prover to demonstrate that $f(X)$ and $\widetilde{f}(X)$ are linked by the permutation σ.

The Prover constructs $P(X) = \sum_{i=0}^{n} e_i \cdot X^i$ as follows:

- $e_0 = 1$.
- For every $i \in [n-1]$, $e_{i+1} = e_i \cdot \frac{[c_i + \beta \cdot i + \gamma]}{[\widetilde{c}_i + \beta \cdot \sigma(i) + \gamma]}$.

Note that, by construction,

$$\begin{aligned} e_n \cdot \frac{[c_n + \beta \cdot n + \gamma]}{[\widetilde{c}_n + \beta \cdot \sigma(n) + \gamma]} \;&=\; \prod_{i=0}^{n} \frac{[c_i + \beta \cdot i + \gamma]}{[\widetilde{c}_i + \beta \cdot \sigma(i) + \gamma]} \\ &=\; 1 \;=\; e_0. \end{aligned}$$

The Prover sends a commitment to $P(X)$ and to its right-shift

$$\texttt{RShift}(P(X)) := X \cdot P(X) \pmod{X^{n+1} - 1}$$

along with a proof that the latter is a right-shift of the former. He sends a zero-knowledge proof that $P(0) = 1$, which is equivalent to the constant term of $P(X)$ being 1.

The Prover now uses the Hadamard product protocol (`ZKPoHadProd`) to show that

$$f_{\beta,\gamma}(X) \;\odot\; P(X) \;=\; \widetilde{f}_{\beta,\gamma}(X) \;\odot\; \texttt{RShift}(P(X)).$$

The last equation implies that

$$\prod_{i=0}^{n} (c_i + \beta \cdot i + \gamma) = \prod_{i=0}^{n} (\widetilde{c}_i + \beta \cdot \sigma(i) + \gamma).$$

Since the challenges β, γ were randomly and uniformly generated, the following lemma implies that with overwhelming probability, $\widetilde{c}_i = c_{\sigma(i)}$ for each i.

Lemma 7.1. *Let $c_0, \cdots, c_n$, $\widetilde{c}_0, \cdots, \widetilde{c}_n$ be elements of $\mathbb{F}_p$. Let σ be a permutation of $[n]$. For random elements $\beta, \gamma \in \mathbb{F}_p$, if*

$$\prod_{i=0}^{n} (c_i + \beta \cdot i + \gamma) \;=\; \prod_{i=0}^{n} (\widetilde{c}_i + \beta \cdot \sigma(i) + \gamma) \;\;,$$

then with overwhelming probability, $\widetilde{c}_i = c_{\sigma(i)}$ for every i.

A proof of this lemma can be found in the appendix of [GWC19]. It hinges on the Schwartz-Zippel lemma.

7.2 Custom gates

A well-known feature of PLONK is that it allows for optimized custom gates in addition to addition, multiplication and Boolean gates. For instance, the circuit for the MiMc hash function benefits greatly from a gate for the cubic function. To this end, we describe a protocol that will allow for certain optimized gates in a setting where PLONK is instantiated via coefficients rather than Lagrange bases.

Let $f(X)$, $\widetilde{f}(X)$ be committed polynomials. Let $P(X)$ be a publicly known polynomial. Let $\mathcal{I}$ be a index set. We describe a protocol that allows a Prover to succinctly demonstrate that the coefficients of $f(X)$, $\widetilde{f}(X)$ are linked as follows:

$$\forall\, i\ \in \mathcal{I},\ \ P\big(\mathtt{Coef}(f\ ,\ i)\big) = \mathtt{Coef}(\widetilde{f}\ ,\ i).$$

In other words, there exist $c_i \in \mathbb{F}_p$ $(i \in \mathcal{I})$ such that

$$f(X) \odot \chi_{\mathcal{I}}(X) = \sum_{i\in\mathcal{I}} c_i X^i\ \ ,\ \ \widetilde{f}(X) \odot \chi_{\mathcal{I}}(X) = \sum_{i\in\mathcal{I}} P(c_i)\cdot X^i.$$

We assume the maximum element of $\mathcal{I}$ is such that the product

$$\max(\mathcal{I}) \cdot 2(\deg(P)+1)$$

is less that the length of the CRS. In other words, a Prover should be able to commit to a polynomial of this degree. This is a HVZK for the relation:

$$\mathcal{R}_{\mathtt{CoefRel}}[g_1,\ (a,\ \widetilde{a},\ \mathtt{C}_{\mathcal{I}}),\ P(X)] = \left\{ \begin{array}{l} \big((a,\ \widetilde{a},\ \mathtt{C}_{\mathcal{I}}\ \in\ \mathbb{G}_1,\ P(X) \in \mathbb{F}_p[X]), \\ f(X),\ \widetilde{f}(X)\ \in \mathbb{F}_p\ ,\ \mathcal{I} \subseteq [\deg(f)]\big) : \\ g_1^{f(s)} = a\ ,\ g_1^{\widetilde{f}(s)} = \widetilde{a}\ ,\ g_1^{\chi_{\mathcal{I}}(s)} = \mathtt{C}_{\mathcal{I}} \\ P(\mathtt{Coef}(f\ ,\ i)) = \mathtt{Coef}(\widetilde{f}\ ,\ i)\ \forall\ i \in \mathcal{I} \end{array} \right\}$$

We fix a public integer k larger than the degree of $P(X)$. For brevity we denote the coefficients of $f(X)$ by c_i and those of $\widetilde{f}(X)$ by $\widetilde{c}_i$.

Our protocol hinges on the simple observation that the following are equivalent:

1. For every i in $\mathcal{I}$, $P(c_i) = \widetilde{c}_i$
2. For every i in $\mathcal{I}$, $P(X) \equiv \widetilde{c}_i \pmod{X - c_i}$
3. There exists a polynomial $\widehat{f}(X)$ with the decomposition

$$\widehat{f}(X) = \sum_{i\in\mathcal{I}} \widehat{f}_i(X)\cdot x^{k\cdot(\max(\mathcal{I})-i)}\ \ ,\ \ \deg(\widehat{f}_i) < k\ \forall\ i$$

such that for a randomly generated challenge $\beta \in \mathbb{F}_p$,

$$\sum_{i\in\mathcal{I}} \widehat{f}_i(X)\cdot(X-c_i)\cdot\beta^i = \sum_{i\in\mathcal{I}} [P(X) - \widetilde{c}_i]\cdot\beta^i$$

The last sum can be simplified as follows:

$$\sum_{i\in\mathcal{I}} [P(X) - \widetilde{c}_i]\cdot\beta^i = [P(X)\cdot\chi_{\mathcal{I}}(\beta)] - \widetilde{f}_{\mathcal{I}}(\beta),$$

where $\widetilde{f}_{\mathcal{I}}(X) := \widetilde{f}(X) \odot \chi_{\mathcal{I}}(X)$.

Description of the protocol: The Prover starts out by verifiably sending commitments to the polynomials

$$f_{\mathcal{I}}(X) := f(X) \odot \chi_{\mathcal{I}}(X) \ , \ \widetilde{f}_{\mathcal{I}}(X) := \widetilde{f}(X) \odot \chi_{\mathcal{I}}(X)$$

using the protocol `ZKHadPoProd`. The Prover also sends a commitment to the polynomial

$$X \cdot \chi_{\mathcal{I}}(X^k) - f_{\mathcal{I}}(X^k) = \sum_{i \in \mathcal{I}} (X - c_i) \cdot X^{k \cdot i}.$$

The Prover computes the polynomial

$$\begin{aligned} \widehat{f}_{\mathcal{I}}(X) &:= \sum_{i \in \mathcal{I}} \frac{P(X) - P(c_i)}{X - c_i} \cdot X^{k \cdot [\max(\mathcal{I}) - i]} \\ &= \sum_{i \in \mathcal{I}} \frac{P(X) - \widetilde{c}_i}{X - c_i} \cdot X^{k \cdot [\max(\mathcal{I}) - i]} \end{aligned}$$

and sends a commitment to this polynomial.

In response to a randomly generated challenge β, the Prover computes the polynomial

$$\beta \cdot X \cdot \chi_{\mathcal{I}}(\beta \cdot X^k) - f_{\mathcal{I}}(\beta \cdot X^k) = \sum_{i \in \mathcal{I}} (X - c_i) \cdot \beta^i \cdot X^{k \cdot i}$$

and verifiably sends a commitment to this polynomial using the protocol `ZKPoTwist`.

The product

$$f_{\Pi,\beta}(X) := \widehat{f}(X) \cdot \big[\beta \cdot X \cdot \chi_{\mathcal{I}}(\beta \cdot X^k) - f_{\mathcal{I}}(\beta \cdot X^k)\big]$$

is a polynomial of degree $2k \cdot \max(\mathcal{I})$. The Prover verifiably sends a commitment to this product using the protocol `ZKPoProd`.

We now use a trick similar to the one we used for the Hadamard product protocol. Instead of exploiting the middle coefficient of the product, we exploit the subvector formed by the coefficients at positions from $X^{k \cdot \max(\mathcal{I})}$ to $X^{k \cdot [1 + \max(\mathcal{I})] - 1}$ (inclusive).

Let $\widetilde{\mathcal{I}}$ denote the index set given by the interval

$$\widetilde{\mathcal{I}} = \big[k \cdot \max(\mathcal{I}),\ k \cdot \big(1 + \max(\mathcal{I})\big) - 1\big].$$

Its characteristic polynomial is given by

$$\chi_{\widetilde{\mathcal{I}}}(X) := \sum_{j = k \cdot \max(\mathcal{I})}^{k \cdot (\max(\mathcal{I}) + 1) - 1} X^j = X^{k \cdot \max(\mathcal{I})} \cdot \frac{X^k - 1}{X - 1}.$$

We have

$$\begin{aligned} f_{\Pi,\beta}(X) \ \odot \ \chi_{\widetilde{\mathcal{I}}}(X) &= X^{k \cdot \max(\mathcal{I})} \cdot \sum_{i \in \mathcal{I}} \big[P(X) - P(c_i)\big] \cdot \beta^i \\ &= X^{k \cdot \max(\mathcal{I})} \cdot \sum_{i \in \mathcal{I}} \big[P(X) - \widetilde{c}_i)\big] \cdot \beta^i \\ &= X^{k \cdot \max(\mathcal{I})} \cdot [P(X) \cdot \chi_{\mathcal{I}}(\beta) - \widetilde{f}_{\mathcal{I}}(\beta)]. \end{aligned}$$

The Prover uses the protocol `ZKHadPoProd` to succinctly demonstrate this relation. This implies that the polynomials

$$\widehat{P}_i(X) := \mathtt{Coef}_k(\widehat{f}\ ,\ \max(\mathcal{I}) - i) \qquad (i \in \mathcal{I})$$

satisfy the equation

$$\sum_{i \in \mathcal{I}} \widehat{P}_i(X) \cdot (X - c_i) \cdot \beta^i = \sum_{i \in \mathcal{I}} [P(X) - \widetilde{c}_i] \cdot \beta^i.$$

Since the element β was randomly generated, this implies that with overwhelming probability,

$$\widehat{P}_i(X) \cdot (X - c_i) = P(X) - \widetilde{c}_i \ \ \forall\, i \in \mathcal{I}.$$

In particular, $P(X) \equiv \widetilde{c}_i \pmod{X - c_i}$ and hence, $\widetilde{c}_i = P(c_i)$ for every $i \in \mathcal{I}$

References

[ABC+12] J.H. Ahn, D. Boneh, J. Camenisch, S. Hohenberger, A. Shelat and B. Waters. *Computing on authenticated data*. In Ronald Cramer, editor, TCC 2012, volume 7194 of LNCS, pages 1-20.Springer, Heidelberg, March 2012.

[BBF19] D. Boneh, B. Bunz, B. Fisch, *Batching Techniques for Accumulators with Applications to IOPs and Stateless Blockchains*, In Alexandra Boldyreva and Daniele Micciancio, editors, CRYPTO 2019, Part I, volume 11692 of LNCS, pages 561–586. Springer, Heidelberg, August 2019.

[BCKL21] E. Ben-Sasson, D. Carmon, S. Kopparty, D. Levit, *Elliptic Curve Fast Fourier Transform* (*ECFFT*) *Part I: Fast Polynomial Algorithms over all Finite Fields*, https://arxiv.org/abs/2107.08473

[BGG17] S. Bowe, A. Gabizon, M. Green, *A multi-party protocol for constructing the public parameters of the Pinocchio zk-SNARK*

[CF13] D. Catalano, D. Fiore, *Vector commitments and their applications*, In Kaoru Kurosawa and Goichiro Hanaoka, editors, PKC 2013, volume 7778 of LNCS, pages 55–72. Springer, Heidelberg, February / March 2013

[Ch10] J.H.Cheon, *Discrete Logarithm Problems with Auxiliary Inputs*

[CL02] J. Camenisch and A. Lysyanskaya. *Dynamic accumulators and application to efficient revocation of anonymous credentials*. In Moti Yung, editor, CRYPTO 2002, volume 2442 of LNCS, pages 61-76. Springer, Heidelberg, August 2002.

[CPZ18] A. Chepurnoy, C. Papamanthou, Y. Zhang, *EDRAX : A Cryptocurrency with Stateless Transaction Validation*, Cryptology ePrint Archive, Report 2018/968, 2018. https://eprint.iacr.org/2018/968.

[DT08] I. Damgard, N. Triandopolous, *Supporting Non-membership Proofs with Bilinear-map Accumulators*, Cryptology ePrint Archive, Report 2008/538, 2008. http://eprint.iacr.org/2008/538.

[FVY14] C. Fromknecht, D. Velicanu, and S. Yakoubov. *A decentralized public key infrastructure with identity retention*. Cryptology ePrint Archive, Report 2014/803, 2014. http://eprint.iacr.org/2014/803.

[FST06] D. Freeman, M. Scott, E. Teske, *A taxonomy of pairing-friendly elliptic curves*

[FS87] A. Fiat, A. Shamir, *How to prove yourself: Practical solutions to identification and signature problems*. In Andrew M. Odlyzko, editor, CRYPTO'86, volume 263 of LNCS, pages 186–194. Springer, Heidelberg, August 1987

[GWC19] A. Gabizon, Z. Williamson, O. Ciobotoru *PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge*, https://eprint.iacr.org/2019/953,

[GGM14] C. Garman, M. Green, and I. Miers. *Decentralized anonymous credentials*. In NDSS 2014.

[KS98] E. Kaltofen and V. Shoup. *Subquadratic-time factoring of polynomials over finite fields*. Mathematics of computation, 67(223):1179–1197, 1998

[KZG10] A. Kate, G. Zaverucha, and I. Goldberg. *Constant-size commitments to polynomials and their applications*. In Masayuki Abe, editor, ASIACRYPT 2010, volume 6477 of LNCS, pages 177–194. Springer, Heidelberg, December 2010.

[KT07] K. Karabina, E. Teske *On prime-order elliptic curves with embedding degrees* 3, 4 *and* 6

[LM18] R. Lai, G. Malavolta, *Optimal succinct arguments via hidden order groups*, Cryptology ePrint Archive, Report 2018/705, 2018. https://eprint.iacr.org/2018/705

[Mil86] V. Miller, *Short Programs for functions on Curves*

[MGGR13a] I. Miers, C. Garman, M. Green, and A. D. Rubin. *Zerocoin: Anonymous distributed E-cash from Bitcoin*. In 2013 IEEE Symposium on Security and Privacy, pages 397-411. IEEE Computer Society Press, May 2013

[Ngu05] L. Nguyen, *Accumulators from bilinear pairings and applications*, CT-RSA, 3376:275–292, 2005

[PST13] C. Papamanthou, E. Shi and R. Tamassia, *Signatures of correct computation*, in: Theory of Cryptography 2013, Lecture Notes in Comput. Sci. 7785, Springer, Heidelberg (2013), 222–242.

[Sla12] Daniel Slamanig. *Dynamic accumulator based discretionary access control for outsourced storage with unlinkable access* - (short paper). In Angelos D. Keromytis, editor, FC 2012, volume 7397 of LNCS, pages 215-222. Springer, Heidelberg, February / March 2012.

[STS99b] Tomas Sander and Amnon Ta-Shma. *Flow control: A new approach for anonymity control in electronic*

cash systems. In Matthew Franklin, editor, FC'99, volume 1648 of LNCS, pages 46-61. Springer, Heidelberg, February 1999

[Th19] S. Thakur, *Batching non-membership proofs with bilinear accumulators*,https://eprint.iacr.org/2019/1147

[Wes18] B. Wesolowski, *Efficient verifiable delay functions*, In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pages 379–407. Springer, 2019.

A List of protocols and the relations they are HVZKs for

1. `PoKE`* (*Proof of knowledge of the exponent with base* g_1

$$\mathcal{R}_{\texttt{KE}^*}[g_1,\ a] = \{(a \in \mathbb{G}_1),\ f(X) \in \mathbb{F}_p[X]) : g_1^{f(s)} = a\}$$

2. `PoKE` (*Proof of knowledge of the exponent*)

$$\mathcal{R}_{\texttt{KE}}[a,\ b] = \{((a,b) \in \mathbb{G}_1^2),\ f(X) \in \mathbb{F}_p[X]) : a^{f(s)} = b\}$$

3. `ZKPoDer` (*Proof of derivative*)

$$\mathcal{R}_{\texttt{Der}}[g_1,\ (a,b)] = \{(a, b \in \mathbb{G}_1),\ f(X) \in \mathbb{F}_p[X]) : g_1^{f(s)} = a\ ,\ g_1^{f'(s)} = b\}$$

4. `ZKPoGCD` (*Proof of GCD*)

$$\mathcal{R}_{\texttt{GCD}}[g_1,\ (a_1,\ a_2),\ a_{1,2}] = \left\{ \begin{array}{l} \big((a_1,\ a_2,\ a_{1,2}\ \in \mathbb{G}_1) \\ f_1(X),\ f_2(X)\ f_{1,2}(X)\ \in\ \mathbb{F}_p[X]\big) : \\ g_1^{f_1(s)} = a_1\ ,\ g_1^{f_2(s)} = a_2,\ g_1^{f_{1,2}(s)} = a_{1,2} \\ \gcd(f_1(X),\ f_2(X)) = f_{1,2}(X) \end{array} \right\}$$

5. `ZKPoSep` (*Proof of separable polynomial commitment*)

$$\mathcal{R}_{\texttt{Sep}}[g_1,\ a] = \{(a, b \in \mathbb{G}_1),\ f(X) \in \mathbb{F}_p[X]) : g_1^{f(s)} = a\ ,\ \mathbf{gcd}(f(X), f'(X)) = 1\}$$

6. `ZKPoProd` (*Proof of product*)

$$\mathcal{R}_{\texttt{Prod}}[g_1,\ (a_1,a_2),\ a_{1,2}] = \left\{ \begin{array}{l} (a_1, a_2 \in \mathbb{G}_1),\ f_1(X), f_2(X) \in \mathbb{F}_p[X]) : \\ g_1^{f_1(s)} = a_1\ ,\ g_1^{f_2(s)} = a_2\ ,\ g_1^{f_1(s)\cdot f_2(s)} = a_{1,2} \end{array} \right\}$$

7. `ZKPoTwist` (*Proof of twist*)

$$\mathcal{R}_{\texttt{Twist}}[g_1,\ (a,\gamma),\ a_\gamma] = \left\{ \begin{array}{l} \big((a, a_\gamma \in \mathbb{G}_1\ ,\ \gamma \in \mathbb{F}_p),\ f(X) \in \mathbb{F}_p[X]\big) : \\ g_1^{f(s)} = a\ ,\ g_1^{f(\gamma\cdot s)} = a_\gamma \end{array} \right\}$$

8. `ZKPoComp` (*Proof of composition*)

$$\mathcal{R}_{\texttt{Comp}}[g_1,\ (a_1,a_2),\ a_{1,2}] = \left\{ \begin{array}{l} (a_1, a_2 \in \mathbb{G}_1),\ f_1(X), f_2(X) \in \mathbb{F}_p[X]) : \\ g_1^{f_1(s)} = a_1\ ,\ g_1^{f_2(s)} = a_2\ ,\ g_1^{f_1((f_2(s))} = a_{1,2} \end{array} \right\}$$

9. `ZKPoRev` (*Proof of reverse polynomial*)

$$\mathcal{R}_{\texttt{Rev}}[g_1,\ (a_1,a_2)] = \{(a_1, a_2 \in \mathbb{G}_1),\ f(X) \in \mathbb{F}_p[X]) : g_1^{f(s)} = a_1\ ,\ g_1^{s^{\deg(f)}\cdot f(s^{-1})} = a_2\}$$

10. `ZKPoDegMono` (*Proof of degree of monomial*)

$$\mathcal{R}_{\texttt{DegMono}}[g_1,\ (a,a_{\deg})] = \{(a, a_{\deg} \in \mathbb{G}_1),\ n \in \mathbb{Z}^{\geq 0},\ c \in \mathbb{F}_p) :\ g_1^{c\cdot s^n} = a\ ,\ g_1^n = a_{\deg}\}$$

11. `ZKPoDegUp` (*Proof of degree upper bound*)

$$\mathcal{R}_{\texttt{Deg}}[g_1,\ (a_1,a_2)] = \{(a_1, a_2 \in \mathbb{G}_1),\ f(X) \in \mathbb{F}_p[X],\ n \in \mathbb{Z}) : g_1^{f(s)} = a_1\ ,\ g_1^n = a_2\ ,\ \deg(f) < n\}$$

12. `ZKPoDeg` (*Proof of degree*)

$$\mathcal{R}_{\texttt{Deg}}[g_1,\ (a,a_{\deg})] = \{(a, a_{\deg} \in \mathbb{G}_1),\ f(X) \in \mathbb{F}_p[X]) : g_1^{f(s)} = a\ ,\ g_1^{\deg(f)} = a_{\deg}\}$$

13. `ZKPoCoef` (*Proof of coefficient*)

$$\mathcal{R}_{\texttt{Coef}}[g_1,\ (a,\ a_{\texttt{Pos}},\ a_{\texttt{Coef}})] = \left\{ \begin{array}{l} \big((a,\ a_{\texttt{Pos}},\ a_{\texttt{Coef}}\ \in\ \mathbb{G}_1), \\ f(X),\ f_-(X),\ f_+(X) \in \mathbb{F}_p[X]\ ,\ n \in \mathbb{Z}^{\geq 0}\ ,\ \alpha \in \mathbb{F}_p\big) : \\ g_1^{f(s)} = a\ ,\ g_1^n = a_{\texttt{Pos}}\ ,\ g_1^\alpha = a_{\texttt{Coef}} \\ f(X) = f_-(X) + \alpha X^n + f_+(X) \\ \deg(f_-(X)) < n\ ,\ f_+(X) \equiv 0 \pmod{X^{n+1}} \end{array} \right\}$$

14. `ZKPoDotProd` (*Proof of dot product*)

$$\mathcal{R}_{\texttt{DotProd}}[g_1,\ (a_1,a_2),\ a_{1,2}] = \left\{ \begin{array}{l} (a_1, a_2 \in \mathbb{G}_1),\ f_1(X),\ f_2(X)\ \in\ \mathbb{F}_p[X]) : \\ g_1^{f_1(s)} = a_1\ ,\ g_1^{f_2(s)} = a_2\ ,\ g_1^{f_1\ \circ\ f_2} = a_{1,2} \end{array} \right\}$$

($\circ$ denotes the dot product of vectors/polynomials)

15. `ZKPoHadProd` (*Proof of Hadamard product*)

$$\mathcal{R}_{\texttt{HadProd}}[g_1,\ (a_1,a_2),\ a_{1,2}] = \left\{ \begin{array}{l} \big((a_1, a_2, a_{1,2} \in \mathbb{G}_1),\ f_1(X), f_2(X) \in \mathbb{F}_p[X]\big) : \\ g_1^{f_1(s)} = a_1\ ,\ g_1^{f_2(s)} = a_2\ ,\ g_1^{f_1 \odot f_2(s)} = a_{1,2} \end{array} \right\}$$

($\odot$ denotes the Hadamard product of vectors/polynomials).

16. `ZKPoBinary` (*Proof of binary polynomial*)

$$\mathcal{R}_{\texttt{Binary}}[g_1,\ a)] = \left\{ \begin{array}{l} \big((a \in \mathbb{G}_1) :\ f(X) \in \mathbb{F}_p[X]\big) : \\ g_1^{f(s)} = a \\ \texttt{Coef}(f(X)\ ,\ i) \in \{0,1\}\ \ \forall\ i \in [\deg(f)] \end{array} \right\}$$

17. `ZKPoNZCoef` (*Proof of non-zero coefficients*)

$$\mathcal{R}_{\texttt{NZCoef}}[g_1,\ a,\ a_{\texttt{nz}}] = \left\{ \begin{array}{l} \big((a,\ a_{\texttt{nz}} \in \mathbb{G}_1), \\ f(X) \in \mathbb{F}_p[X]\ ,\ \mathcal{I} \subseteq [\deg(f)]\big) : \\ g_1^{f(s)} = a\ ,\ g_1^{\chi_{\mathcal{I}}(s)} = a_{\texttt{nz}}, \\ \forall\ i \in \mathcal{I},\ \texttt{Coef}(f(X)\ ,\ i) \neq 0 \end{array} \right\}$$

($\texttt{Coef}(f(X)\ ,\ i)$ denotes the coefficient of $f(X)$ at X^i. $\chi_{\mathcal{I}}(X)$ is the polynomial $\sum\limits_{i \in \mathcal{I}} X^i$.)

18. `ZKPoCoefRel` (*Protocol for coefficient relation*)

$$\mathcal{R}_{\texttt{CoefRel}}[g_1,\ (a,\ \widetilde{a},\ \texttt{C}_{\mathcal{I}}),\ P(X)] = \left\{ \begin{array}{l} \big((a,\ \widetilde{a},\ \texttt{C}_{\mathcal{I}} \in \mathbb{G}_1,\ P(X) \in \mathbb{F}_p[X]), \\ f(X),\ \widetilde{f}(X) \in \mathbb{F}_p\ ,\ \mathcal{I} \subseteq [\deg(f)]\big) : \\ g_1^{f(s)} = a\ ,\ g_1^{\widetilde{f}(s)} = \widetilde{a}\ ,\ g_1^{\chi_{\mathcal{I}}(s)} = \texttt{C}_{\mathcal{I}} \\ P(\texttt{Coef}(f\ ,\ i)) = \texttt{Coef}(\widetilde{f}\ ,\ i)\ \forall\ i \in \mathcal{I} \end{array} \right\}$$

19. `ZKPoCoefMulSet` (*Protocol for coefficient multiset*)

$$\mathcal{R}_{\texttt{CoefMulSet}}[g_1,\ (a,\ A)] = \left\{ \begin{array}{l} \big((a,\ A \in \mathbb{G}_1), \\ n \leq \big|\texttt{CRS}\big| - 1\ \ ,\ \ c_0, \cdots, c_n \in \mathbb{F}_p\big) : \\ a = g_1^{\sum\limits_{i=0}^{n} c_i \cdot s^i}\ \ ,\ \ A = g_1^{\prod\limits_{i=0}^{n} (s + c_i)} \end{array} \right\}$$

20. `ZKPoPerm` (*Protocol for permutation*)

$$\mathcal{R}_{\texttt{Perm}}[g_1,\ (a,\ b)] = \left\{ \begin{array}{l} \big((a,\ b \in \mathbb{G}_1)\ \ ,\ \ f(X) \in \mathbb{F}_p[X] \\ \text{Permutation } \sigma :\ \big[|\texttt{CRS}|\big] \longrightarrow \big[|\texttt{CRS}|\big]\big) : \\ a = g_1^{f(s)}\ \ ,\ \ b = g_1^{f^{\sigma}(s)} \end{array} \right\}$$

21. `ZKPoPerm`$\uparrow$ (*Protocol for increasing permutation*)

$$\mathcal{R}_{\texttt{Perm}\uparrow}[g_1,\ (a,\ b),\ C_{\mathcal{I}}] = \left\{ \begin{array}{l} \big((a,\ b,\ C_{\mathcal{I}} \in \mathbb{G}_1), \\ f(X) \in \mathbb{F}_p[X]\ \ ,\ \ \mathcal{I} \subseteq [|\texttt{CRS}| - 1] \\ \text{Permutation } \sigma :\ \big[|\texttt{CRS}|\big] \longrightarrow \big[|\texttt{CRS}|\big]\big) : \\ a = g_1^{f(s)}\ \ ,\ \ b = g_1^{f^{\sigma}(s)}\ \ ,\ \ C_{\mathcal{I}} = g_1^{\chi_{\mathcal{I}}(s)} \\ \forall\ i, j \in \mathcal{I},\ i < j \iff \sigma(i) < \sigma(j) \end{array} \right\}$$

22. `ZKPoSubseq` (*Protocol for subsequence*)

$$\mathcal{R}_{\texttt{Subseq}}[g_1,\ (a,\ b),\ (C_{\mathcal{I}}.\ C_{\mathcal{J}})] = \left\{ \begin{array}{l} \big((a,\ b,\ C_{\mathcal{I}},\ C_{\mathcal{J}} \in \mathbb{G}_1), \\ f(X),\ \widetilde{f}(X) \in \mathbb{F}_p[X]\ \ ,\ \mathcal{I}, \mathcal{J} \subseteq [|\texttt{CRS}| - 1] \\ \text{Permutation } \sigma :\ [\max(\mathcal{I} \cup \mathcal{J})] \longrightarrow [\max(\mathcal{I} \cup \mathcal{J})]\big) : \\ a = g_1^{f(s)}\ \ ,\ \ b = g_1^{\widetilde{f}(s)}\ \ ,\ \ C_{\mathcal{I}} = g_1^{\chi_{\mathcal{I}}(s)}\ \ ,\ \ C_{\mathcal{J}} = g_1^{\chi_{\mathcal{J}}(s)} \\ \forall\ i, j \in \mathcal{I},\ i < j \iff \sigma(i) < \sigma(j) \end{array} \right\}$$