UJ crypto course: the KZG PCS scheme and PlonK SNARK

Ariel Gabizon
Zeta Function Technologies

June 14, 2024

1 The KZG polynomial commitment scheme - an informal intro

The idea of polynomial commitment schemes is that a prover **P** can send a short commitment to a large polynomial $f \in \mathbb{F}_{<d}[X]$; and later open it at a point $z \in \mathbb{F}$ chosen by the verifier. **P** can also construct a proof π that the value he sends is really $f(z)$ for the f he had in mind during commitment time.

2 The KZG commitment scheme:

Prequisites: Given integer security parameter λ. We assume access to

- Groups $\mathbb{G}$,$\mathbb{G}_t$ of prime order r written additively with $\lambda^{\omega(1)} < r \leq 2^\lambda$
- randomly chosen generator $g \in \mathbb{G}$ and generator $g_t \in \mathbb{G}_t$.
- Denote by $\mathbb{F}$ the field of size r. A bi-linear pairing function $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_t$ such that $\forall a, b \in \mathbb{F}$, $e(a\cdot, b \cdot g) = g_t^{a \cdot b}$

  For $c \in \mathbb{F}$, we use the notation $[c] := c \cdot g$.

Remark 2.1. *How do we actually construct pairings?? They were first constructed by André Weil, where $\mathbb{G}$ is taken to be the degree zero divisor class group of an algebraic curve - which is the same as the so-called Jacobian of the curve. In crypto we always use elliptic curves which are a special case, and where this group is isomorphic to the curve itself! These pairings were part of Weil's proof of what's called the Riemann hypothesis for curves over finite fields/Algebraic function fields. Much later, Victor Miller showed a practical algorithm to compute them (the "Miller loop").*

2.1 The scheme:

- **setup:**: Generate $\mathsf{srs} = [1], [x], \ldots, [x^d]$, for random $x \in \mathbb{F}$.
- **Commitment to $f \in \mathbb{F}_{<d}[X]$:**
  $\mathsf{cm}(f) := [f(x)]$
- The $\mathsf{open}(\mathsf{cm}, z, s; f)$ protocol - proving $f(z) = s$
  1. **P** computes $h(X) := \frac{f(X)-f(z)}{X-z}$
  2. **P** sends $\pi = [h(x)]$.
  3. **V** outputs acc if and only if

$$e(\mathsf{cm} - [s], [1]) = e(\pi, [x - z])$$

3 The Algebraic Group Model

We wish to capture the notion of an Algebraic Adversary that can generate new group elements by doing "natural" operations on the set of group elements he already received.

The intuition is that when the discrete log is hard, the group elements look random to (an efficient) adversary, and thus there's no other way for him to produce "useful" group elements.

More formally,

Definition 3.1. *By an* SRS-based protocol *we mean a protocol between a prover* **P** *and verifier* **V** *such that at before the protocol begins a randomized procedure* setup *is run, returning a string* $\mathsf{srs} \in \mathbb{G}^n$.

In our protocols, by an algebraic adversary $\mathcal{A}$ *in an SRS-based protocol we mean a* $\mathsf{poly}(\lambda)$*-time algorithm which satisfies the following.*

- *Whenever $\mathcal{A}$ outputs an element $A \in \mathbb{G}$, it also outputs a vector v over $\mathbb{F}$ such that $A = \sum_{i \in [n]} v_i \mathsf{srs}_i$.*

4 Knowledge Soundness of the KZG scheme

We say a PCS has *Knowledge soundness in the Algebraic Group Model* if, there exists an efficient algorithm E such that any algebraic adversary $\mathcal{A}$ has probability at most $\mathsf{negl}(\lambda)$ to win the following game:

1. $\mathcal{A}$ outputs a commitment cm.
2. E outputs a polynomial $f \in \mathbb{F}_{<d}[X]$

3. $\mathcal{A}$ outputs $z, s \in \mathbb{F}, \pi \in \mathbb{G}$.

4. $\mathcal{A}$ takes part of $\mathbf{P}$ in $\mathsf{open}(\mathsf{cm}, z, s)$.

5. $\mathcal{A}$ wins iff

   (a) $\mathbf{V}$ outputs acc in the end of open.

   (b) $f(z) \neq s$.

We use the following extension of the discrete-log assumption:

Definition 4.1. *The Q-DLOG assumption for $\mathbb{G}$ says that: For any efficient algorithm A, the probability that given $[1], [x], \ldots, \left[x^Q\right]$ for random $x \in \mathbb{F}$ outputs x is $\mathsf{negl}(\lambda)$*

Lemma 4.2. *Assuming the d-DLOG for $\mathbb{G}$, KZG has knowledge soundness in the algebraic group model*

Proof. We must define the extractor E: When $\mathcal{A}$ outputs cm, since it's algebraic, it also outputs $f \in \mathbb{F}_{<d}[X]$ such that $\mathsf{cm} = [f(x)]$. E will simply output f. Let $\mathcal{A}$ be some efficient algebraic adversary.

We will describe an algorithm B following the game between E and $\mathcal{A}$ such that whenever $\mathcal{A}$ wins the game, B finds x!.

During open when $\mathcal{A}$ sends π, it also sends $h \in \mathbb{F}_{<d}[X]$ such that $\pi = [h(x)]$. Assume we're in the case that $\mathcal{A}$ won the game. This means that $\mathcal{A}$ sent z, s and π such that $\mathbf{V}(\mathsf{cm}, z, s, \pi) = \mathsf{acc}$. This means that $e(\mathsf{cm} - [s], [1]) = e(\pi, [x - z])$ In our case this is the same as $e([f(x)] - [s], [1]) = e([h(x)], [x - z])$ This implies

$$f(x) - s = h(x)(x - z)$$

Define the polynomial

$$p(X) := f(X) - s - h(X)(X - z)$$

We claim that $p(X)$ *can't* be the zero polynomial: If it was, in particular $p(z) = 0$, and then $f(z) - s = h(z)(z - z) = 0$, which means $f(z) = s$ - but $\mathcal{A}$ won the game so $f(z) \neq s$!

So $p(X)$ isn't the zero polynomial in the case that $\mathcal{A}$ won. We claim that in this case we can find x:

Note that we can compute the coefficients of p, since $\mathcal{A}$ sent the coefficients of f and h.

Since $\mathbf{V}$ accepting means that $p(x) = 0$. Thus, we can factor p, and x will be one of its roots. We can check for each root γ of p if $\gamma = x$ by checking $[\gamma] = [x]$.

□

5 Lecture 2: Proving circuit satisfiability Plonk style

We'll look at arithmetic circuits C over $\mathbb{F}$

- with addition and multiplication gates of fan-in 2 and unbounded fan-out.
- We'll assume there's a unique output wire
- we'll denote by n the number of gates and m the number of wires.
- We'll assume input wires go only into one gate (as someone astutely observed during the lecture, otherwise we need extra copy constraints in the BP program described later).

Draw example of $(a + b) \cdot c$ circuit

Given such a circuit we can look at the relation $\mathcal{R}_C$ containing all pairs $(z \in \mathbb{F}, \omega \in \mathbb{F}^m)$ such that w is a valid assignment to the wires of C with output value $\omega_m = z$.

Though in SNARK context, we almost always want to look at relations and separate the instance and witness, for simplicity we'll focus on simply proving knowledge of *some* assignment to C.

5.1 Baby-Plonk programs

We wish to reduce checking an assignment to a circuit to checking an assignment to a Plonk program. In the literature you will find references to Turbo-plonk and ultra-plonk programs. Here for simplicity, we look at a much more restricted notion of "baby plonk" programs, that are still sufficient to capture our circuits.

Definition 5.1. *A* Baby-PlonK *program* BP *is defined by*

1. *vectors* $\mathrm{A}, \mathrm{M} \in \mathbb{F}^n$.
2. *A set of "copy constraints" of the form "$w_{i,j} = w_{i',j'}$" for some $i, i' \in \{1, 2, 3\}$ and $j, j' \in [n]$*

A set of vectors $a, b, c \in \mathbb{F}^n$ satisfies BP *if*

1. *For each $i \in [n]$*
$$\mathrm{A}_i \cdot (a_i + b_i) + \mathrm{M}_i(a_i \cdot b_i) = c_i$$
2. *Setting $w_1 = a, w_2 = b, w_3 = c$ all copy constraints are satisfied.*

Draw example as rectangle - emphasizing local vs global

5.2 Some algebra

Let's assume n is a power of two, and n divides $|F| - 1$. That means there's an element $g \in \mathbb{F}$ of order n. That is $H = \{g, \ldots, g^n\}$ is a multiplicative subgroup of order n. We denote by $Z_H(X) = \prod_{i \in [n]}(X - g^i)$ the vanishing polynomial of H. We have $Z_H(X) = X^n - 1$.

We have for any polynomial $F(X)$ that $F(a) = 0$ for all $a \in H$ iff F is divisible by Z_H, i.e. iff there exists $T(X) \in \mathbb{F}[X]$ such that $F(X) = T(X)Z_H(X)$.

Given vector $v \in \mathbb{F}^n$ we'll say a polynomial f *interpolates* v *over* H if it has degree $< n$ for all $i \in [n]$ $f(g^i) = v_i$. There is always such unique $f(X) = \sum v_i L_i(X)$ where $L_i(X) = \frac{g^i}{n}\frac{X^n-1}{X-g^i}$ are the Lagrange base of H.

Given $\mathrm{A}, \mathrm{M}, a, b, c \in \mathbb{F}^n$ we abuse notation and denote by the same names the polynomials interpolating them.

Define

$$F(X) := \mathrm{A}(X) \cdot (a(X) + b(X)) + \mathrm{M}(X)(a(X) \cdot b(X)) - c(X).$$

Assume that a, b, c *satisfy* the copy constraints of BP. (Verifying that is in fact the more interesting part of PlonK and we'll deal with that later!) Then we have that a, b, c satisfy BP iff $F(X)$ is divisible by Z_H.

5.3 A protocol for checking satisfiability (missing the copy constraint checks for now)

Now we can use the KZG scheme to get a protocol checking an assignment for BP. The cool thing is that the proof size will be a *constant* number of $\mathbb{F}$ and $\mathbb{G}$ elements - independent of n! (For construction to work we need that $|\mathbb{G}| = |\mathbb{F}| > n$ so in fact in terms of bit-length the proof is $\Omega(\log n)$)

In the description below we write $\mathsf{com}(()f)$ for the KZG commitment of $f \in \mathbb{F}[X]$.

Below we assume $\mathbf{P}$ has a satisfying assignment (a, b, c) to BP.

Preprocessing: We precompute the KZG commitments $\mathsf{com}(\mathsf{A})$,$\mathsf{com}(\mathsf{M})$ of A, M and send them to $\mathbf{V}$.

The protocol:

1. $\mathbf{P}$ computes and sends $\mathsf{com}(\mathsf{a})$,$\mathsf{com}(\mathsf{b})$,$\mathsf{com}(\mathsf{c})$ to $\mathbf{V}$.
2. With $F(X)$ defined as above, $\mathbf{P}$ computes the quotient polynomial $T(X) := F(X)/Z_H(X)$.
3. $\mathbf{P}$ computes and sends $\mathsf{com}(\mathsf{T})$ to $\mathbf{V}$.
4. $\mathbf{V}$ chooses random $\alpha \in \mathbb{F}$ and sends it to $\mathbf{P}$.
5. $\mathbf{P}$ sends $\bar{A} := A(\alpha), \bar{M} := M(\alpha), \bar{a} := a(\alpha), \bar{b} := b(\alpha), \bar{c} := c(\alpha), \bar{T} := T(\alpha)$ to $\mathbf{V}$.
6. $\mathbf{P}$ sends the KZG opening proofs for the above values.
7. $\mathbf{V}$ verifies the KZG openings proofs for all values.

8. **V** computes $\bar{F} := \bar{A}(\bar{a} + \bar{b}) + \bar{M}\bar{a}\bar{b} - \bar{c}$.

9. **V** accepts iff $\bar{F} = Z_H(\alpha)\bar{T}$.

6 Lecture 3: the PlonK permutation argument (based on Bayer-Groth)

6.1 Copy checks via permutations

Let $v = (a, b, c)$, and redefine n as $n = |v|$. In BP we have a bunch of constraints $v_i = v_j$ for $i, j \in [3n]$. We can reduce all checks to one *permutation check*. This mean checking for some fixed permutation σ on $[n]$, that $\sigma(v) = v$. Here we define $\sigma(v) \in \mathbb{F}^n$ by $\sigma(v)_{\sigma(i)} = v_i \ \forall \ i \in [n]$. **example:** Say we have the constraints $v_1 = v_2, v_2 = v_7, v_5 = v_6$. Define $\sigma = (127)(56)$. Then v satisfies the copy constraints iff $v = \sigma(v)$.

6.2 Mutliset checks via grand products

Let's look at a related problem that we will use soon for the permutation check. Now **P** has *two* vectors f, g of size n and wishes to show to **V** they are equal as multisets, i.e. that there is *some* permutation σ such that $g = \sigma(f)$.

Here is a protocol for this.

1. **V** sends random $\gamma \in \mathbb{F}$

2. **P** shows to **V** that
$$\prod_{i \in [n]} (f_i + \gamma) = \prod_{i \in [n]} (g_i + \gamma)$$

Claim 6.1. *Let $r = |\mathbb{F}|$. The check in equation 2 holds with probability one if f, g are mutliset equal, and probability at most n if they are not.*

Proof. Define the polynomials $F(X) := \prod_{i \in [n]}(f_i + X), G(X) := \prod_{i \in [n]}(g_i + X)$ We are checking $F(\gamma) = G(\gamma)$. If f, g are mutliset equal $F \equiv G$ so this holds for all γ. If they are not $F \not\equiv G$ so they can be equal for at most n $\gamma \in \mathbb{F}$. □

One main question is how **P** efficiently shows to **V** that equation 2 holds when **V** only has commitments to f, g? We will see a solution soon!

6.3 Permuations via mulitset checks

For our SNARK, we wish to check that $g = \sigma(f)$ for a *fixed* permutation σ. We show we can reduce this check to a multiset check.

Claim 6.2. *Given a permutation σ on $[n]$ and vectors $f, g \in \mathbb{F}^n$, we have $g = \sigma(f)$ if and only if the sets of pairs $A := \{(f_i, i)\}_{i \in [n]}$ and $B := \{(g_i, \sigma(i)\}_{i \in [n]}$ are equal as (multi)sets.*

The mulitset protocol dealt with scalars, not tuples. Here we can use a little randomness to reduce the scalars to tuples.

Claim 6.3. *Given sets of pairs A, B as in the claim above and $\beta \in \mathbb{F}$. Define the mutlisets $A' =:= \{f_i + \beta \cdot i\}$, $B' := \{g_i + \beta \cdot \sigma(i)\}$. Then if A, B are different as sets, then A', B' are different as multisets except with probability n/r over β*

Proof. If $A \neq B$, there is some pair $(b_1, \beta b_2) \in B \setminus A$. For any fixed $(a_1, a_2) \in A$ the probability that $a_1 + \beta a_2 = b_1 + \beta b_2$ is at most $1/r$. Now do a union bound over elements of A. □

6.4 grand products via polynomial equations

Now we deal with the following question. Suppose **V** has a KZG-commitment $\mathsf{cm}(f)$ to a polynomial of degree $< n$ (as before think of f as interpolating the vector $f \in \mathbb{F}^n$ over H). Suppose that $\prod_{i \in [n]} f_i = 1$. How can **P** prove this to **V**?

Protocol:

1. **P** interpolates on H the vector Z with values $Z_1 = 1$, $Z_i = \prod_{j<i}(f_j)$ for $i \in \{2, \ldots, n\}$. I.e $Z(g^i) = Z_i \ \forall i \in [n]$.
2. **P** computes and send $\mathsf{com}(\mathsf{Z})$ to **V**.
3. **V** sends random $\alpha \in \mathbb{F}$ to **P**.
4. **P** computes poly:
$$T(X) = \frac{L_1(X)(Z(X) - 1)) + \alpha(Z(g \cdot X) - Z(X)f(X))}{Z_H(X)}$$
5. **V** sends random $\gamma \in \mathbb{F}$ to **P**.
6. **P** sends $\bar{T} := T(\gamma), \bar{f} := f(\gamma), \bar{Z} := Z(\gamma), \bar{Z}_g := Z(g \cdot \gamma)$
7. **P** sends the KZG opening proofs for the above values.
8. **V** verifies the KZG openings proofs for all values.
9. **V** computes $\bar{F} := L_1(\gamma)(Z(\gamma) - 1) + \alpha(Z(g \cdot \gamma) - Z(\gamma)f(\gamma)))$
10. **V** accepts iff $\bar{F} = Z_H(\gamma)\bar{T}$.

Claim 6.4. *If $\prod_{i \in [n]} f_i = \prod_{i \in [n]} f(g^i) \neq 1$ then* **V** *rejects e.w.p $(3n)/r$.*

Proof. Define the polynomial
$$F(X) = L_1(X)(Z(X) - 1) + \alpha(Z(g \cdot X) - Z(X)f(X))$$
similar to the protocol of the previous lecture, the protocol is checking if F vanishes on H via a divisibility check. Assume it does vanish on H. Then e.w.p $1/r$ over α this implies both terms $F_1 = L_1(X)(Z(X) - 1))$ and $F_2 := Z(g \cdot X) - Z(X)f(X)$ vanish on H.

- F_1 vanishing on H implies $Z(g) = 1$.
- F_2 vanishing on H implies that for each $i \in [n]$, $Z(g^{i+1}) = f(g^i)Z(g^i)$. And so
$$1 = Z(g) = Z(g^{n+1}) = \prod_{i \in [n]} f(g^i)$$
as required.

□

6.5 Putting it all together

We sketch how the above components can be used to show $\sigma(v) = v$ assuming $\mathbf{V}$ has $\mathsf{com(v)}$:

$\mathbf{V}$ chooses random $\beta, \gamma \in \mathbb{F}$.

$\mathbf{P}$ uses the above grand product argument to show $\prod_{i \in [n]} f_i = 1$. For the vector f defined as
$$f_i := \frac{v_i + \beta \cdot i + \gamma}{v_i + \beta \cdot \sigma(i) + \gamma}$$

To enable this we compute in a preprocessing phase the polynomials ID, S that interpolate on H the identity and σ permutation. I.e. $ID(g^i) = i, S(g^i) = \sigma(i)$. For full details see Section 5 of the PlonK paper.

7 Lecture 4 - lookup arguments

Say we want to prove in a plonk program that $0 \leq x \leq 2^{s-1}$. We could do it as follows: $\mathbf{P}$ sends the binary decomposition $x_0, \ldots, x_{s-1}$ and proves that

- Send $x_0, \ldots, x_{s-1}$ are binary, via equation $x_i(x_i - 1) = 0$.
- $\sum_{i<s} x_i 2^i = x$

This takes $s + 1$ gates - in fact more like $2s$ since the addition has to be split up. Lookup arguments offer an alternative approach. For polynomials $f, t \in \mathbb{F}[X]$ let's write $f \subset t$ if as *sets* $\{f(a)\}_{a \in H} \subset \{t(a)\}_{a \in H}$. A *lookup protocol* allows proving this to $\mathbf{V}$ having $\mathsf{com(f)}$,$\mathsf{com(t)}$. Say $s = \log n$, $|H| = n$. Say we want to check all of f's vals are $0 \leq f_i \leq 2^s - 1$. In the above example, we could precompute $\mathsf{com(t)}$ for t with values $\{0, \ldots, 2^s - 1\}$ on H and apply lookup protocol.

7.1 plookup - Lookup protocols via multiset checks

Fix $t, f \in \mathbb{F}^n$. Define the *multi-set* $A = \{(f_i, f_i\}_{i \in [n]} \cup \{(t_i, t_{i+1}\}_{i \in [n-1]}$

Claim 7.1. *Assume (for simplicity mainly) t contains distinct values and is sorted: $t_1 < t_2 \ldots < t_n$. f subsett if and only if there exists $s \in \mathbb{F}^{2n-1}$ such that defining $S' = \{(s_i, s_{i+1}\}_{i \in [2n-1]}$, we have $A = S$ as multisets.*

Proof. **only if:** Define s to be the *sorted* version of the (mutli-set) union $f \cup t$. Then $A = S'$. □

7.2 Lookups based on log-derivative

The main claim is :

Claim 7.2. *Assume $r = |\mathbb{F}|$ is prime and larger than n. $f \subset t$ if and only if there exists $m \in \mathbb{F}^n$ such that as rational functions*

$$\sum_{i\in[n]} \frac{m_i}{X + t_i} = \sum_{i\in[n]} \frac{1}{X + f_i}$$

Proof. **only if:** Define m_i to be the number of times t_i appears in f. **if(sketch):** The functions $1/(X + a)$ for different $a \in \mathbb{F}$ are linearly independent, therefore the functions in the equation being equal implies every coefficient is equal. So there can't be any $1/(X + a)$ on the RHS for $a \notin T$. □

protocol idea:check identity on random β

1. **P** computes and send $\mathsf{com}(\mathsf{m})$
2. **V** chooses random $\beta \in \mathbb{F}$.
3. **P** sends $\mathsf{com}(\mathsf{A})$ for polynomial A with $A_i = \frac{m_i}{\beta+t_i} - \frac{1}{\beta+f_i}$
4. **V** checks that A is well-formed and sums to zero on H.

Checking that A is well-formed - has the correct values, can be done by a quotient polynomial similar to others we've seen.

7.3 Proving a polynomial sums to zero:

We rely on the following claim from the Aurora paper:

Claim 7.3. *Given $f \in \mathbb{F}_{<n}[X]$, $\sum_{a\in H} f(a) = (1/n) \cdot f(0)$.*

Proof.

$$f(0) = \sum_{i\in[n]} f_i L_i(0)$$

We have for all $i \in [n]$, $L_i(0) = 1/n$. □