ZK notes

Some keynotes made while learning about ZK, SNARK, STARK and ZK-VM.

Lectures: Youtube Playlist

1 SNARK

SNARK is not necessary ZK and can be quite different from STARK and ZK-VM. But it should be a good starting ground to understand some basic concept.

1.1 Polynomial Commitment Scheme (PCS)

A PCS is a functional commitment for the family $\mathcal{F} \in \mathbb{F}_p^{(\leq d)}[X]$. A prover commits to univariate polynomial f in $\mathbb{F}_p^{(\leq d)}[X]$ and can later prove to the verifier that $v = f(u)$ for public $u, v \in \mathbb{F}_p$.

Some examples PCS (here we focus on KZG'10):

1. Bulletproof (elliptic curves, but verification is $O(d)$)
2. KZG'10 (trusted setup, bilinear), Dory'20 (bilinear)
3. Dark'20 (groups of unknown order)
4. Hash (FRI)

1.2 KZG'10

Set cyclic group $\mathbb{G} = \{0, G, 2 \cdot G, 3 \cdot G, \ldots, (p-1) \cdot G\}$ of order p.

Setup algorithm

1. Sample random $\alpha \in \mathbb{F}_p$.
2. $pp = (H_0 = G, H_1 = \alpha \cdot G, \ldots, H_d = \alpha^d \cdot G) \in \mathbb{G}^{d+1}$.
3. **delete** α (i.e., A trusted setup)

Commitment

In short: $commit(pp, f) \rightarrow com_f$, where $com_f = f(\alpha) \cdot G \in \mathbb{G}$.[1]

Remark. *As a result, the committed message is extremely short (an element G) regardless how large our polynomial is.*

But α is deleted during trusted setup, how does prover compute $f(\alpha)$? Observe:

[1] This is not a hiding commitment

$\Rightarrow f(X) = f_0 + f_1 X + \ldots + f_d X^d$

$\Rightarrow f(\alpha) \cdot G = f_0 \cdot G + f_1 \cdot \alpha \cdot G + \ldots + f_d \cdot \alpha^d \cdot G$

$\Rightarrow f(\alpha) \cdot G = f_0 \cdot H_0 + f_1 \cdot H_1 + \ldots + f_d \cdot H_d$

Evaluation

How to prove $f(u) = v$?

First Observe:

1. If $f(u) = v \iff u$ is a root of polynomial $\hat{f}(X) = f(X) - v$.
2. If u is a root of $\hat{f}(X) \iff \hat{f}(X)$ is divisible by $(x - u)$.
3. $f(u) = v \iff \exists q \in \mathbb{F}_p^{(\leq d)}[X]$ s.t. $q(X)(X - u) = f(X) - v$

The prover then computes quotient polynomial $q(X) = (f(X) - v)/(X - u)$ and sends com_q to verifier.

The verifier accepts if $(\alpha - u) \cdot com_q = com_f - v \cdot G$.

LHS:

$\Rightarrow (\alpha - u) \cdot com_q$

$\Rightarrow (\alpha - u) \cdot (q_0 H_0 + q_1 H_1 + \ldots + q_d H_d)$

$\Rightarrow (\alpha - u) \cdot (q_0 G + q_1 \alpha G + \ldots + q_d \alpha^d G)$

$\Rightarrow commit(pp, (X - u)q(X))$

RHS is similar. [2]

Remark. *The verification work only take constant time, regardless of the degree of the polynomial.*

Extension

1. KZG for k-variant polynomial (PST'13)
2. Batch proofs: prove a batch of commitments in a single step.

[2] Important: verifier does not actually need to know about α. The *pairing* is used here to allow verifier to compute $(\alpha - u) * com_q$ with only G and H_1

1.3 A Useful Observation

A Useful and important observation.

For $0 \neq f \in \mathbb{F}_p^{(\leq d)}[X]$. Let r be a random point $r \leftarrow \mathbb{F}_p$, the probability $pr[f(r) = 0] = d/p$.[3]

For large enough p and reasonable d, e.g., $p \approx 2^{256}$ and $d \leq 2^{40}$, d/p is negligible.

Lemma 1. *for $r \leftarrow \mathbb{F}_p$, if $f(r) = 0$, we can conclude f is identically zero w.h.p.*[4]

Further more, with the same settings.

Lemma 2. *Let $f, g \in \mathbb{F}_p^{(\leq d)}[X]$. For $r \leftarrow \mathbb{F}_p$, if $f(r) = g(r)$ then $f = g$.*

$\Rightarrow f(r) - g(r) = 0$

$\Rightarrow$ Let $h = f - g$, from Lemma 1, h is identical zero w.h.p.

$\Rightarrow f = g$, w.h.p

1.4 Zero Test On H

One of the (and the simplest) poly-IOP tasks that the verifier would like the prover to do.

Let $\omega \in \mathbb{F}_p$ be a primitive k-th root of unity (such that $\omega^k = 1$ and $\omega^n \neq 1$ for $n < k$).

Set $H = \{1, \omega, \omega^2, \ldots, \omega^{k-1}\} \in \mathbb{F}_p$.

Let polynomial $f \in \mathbb{F}_p^{(\leq d)}[X]$.

A zero test is a test from verifier to prover to prove that: f **is identically zero on set** H**.**

Lemma 3. *f is zero on H iff $f(X)$ is divisible by $X^k - 1$.*

1. The prover can compute the quotient polynomial $q(X) = f(X)/(X^k - 1)$. and send the commitment of q to the verifier.
2. The verifier then choose random $r \in \mathbb{F}_p$ and ask prover to open $f(X)$ and $q(X)$ at r.
3. The verifier then accepts the test if $f(r) = q(r) \cdot (r^k - 1)$

As mentioned in Lemma 2, two polynomials that agree on a random point r has a high probability that the two polynomials are identical. Therefore, the above implies $f(X) = q(X)(X^k - 1)$. This proves $f(X)$ is indeed divisible by $X^k - 1$, hence from Lemma 3, f is identical on H.

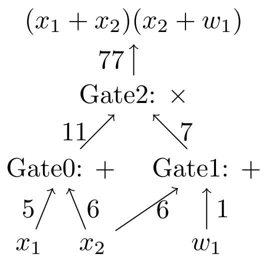
[3] Given f has at most d roots and p elements

[4] Also holds for multivariate polynomial, see SZDL lemma.

1.5 Interpolate Polynomial

Plonk.

1.5.1 Compile a circuit into a computation trace



inputs:	5	6	1
	left	right	out
Gate0	5	6	11
Gate1	6	1	7
Gate2	11	7	77

1.5.2 Encoding the trace as polynomial

$C \leftarrow$: total # of gates

$I \leftarrow |I_x| + |I_w|$: # inputs to circuit

$d \leftarrow 3|C| + |I| = 12$ for our example. (3 since each gate has 3 inputs).

$H \leftarrow \{1, \omega, \ldots, \omega^{(d-1)}\}$

The goal here is to interpolate a polynomial P that encodes the computation trace. To achieve that, we want to

1. let P encodes all inputs, such that $P(\omega^{-j}) =$ inputs # j for all $j = 1, \ldots, |I|$.
2. let P encodes all wires, such that $\forall l = 0, \ldots, |C| - 1$:
   (a) $P(\omega^{3l}) =$ left input of gate # l.
   (b) $P(\omega^{3l+1}) =$ right input of gate # l.
   (c) $P(\omega^{3l+2}) =$ output of gate # l.

This results in 12 constraints for P, which means there exists a P with degree at most 11 that satisfies all the constraints. Prover can then constructs P using Fast Fourier Transform in time $O(d \log d)$, which I don't know how yet.

1.5.3 Prove that encoding is correct

There are four things to prove.

Inputs are correctly encoded.
Both prover and verifier takes input x and interpolate a polynomial $v(X) \in \mathbb{F}_p^{(\leq d)}[X]$ that satisfies $\forall j = 1, \ldots, |I_x| : v(\omega^{-j}) =$input #j.

From the slides, it says construction takes time linear to the size of x, shouldn't it still be using FFT and the time is actually $O(n \log n)$?.

Then prover just proves that $P(y) - v(y) = 0 \;\; \forall y \in H_{inp}$ where H_{inp} is all the input points, i.e., $\{\omega^{-1}, \dots, \omega^{-|I_x|}\}$. This can be done using zero-test.

Gates evaluations are correctly encoded.
Interpolate selector polynomial $S(X) \in \mathbb{F}_p^{(\leq d)}[X]$ such that $\forall l = 0, \dots, |C| - 1$:

1. $S(\omega^{3l}) = 1$ if gate l is addition
2. $S(\omega^{3l}) = 0$ if gate l is multiplication

Observe $\forall y \in H_{gates} = \{1, \omega^3, \omega^6, \dots, \omega^{3(|C|-1)}\}$:

$$S(y) \cdot [P(y) + P(\omega y)] + (1 - S(y)) \cdot P(y) \cdot P(\omega y) = P(\omega^2 y)$$

When $S(y) = 1$, which means a gate is addition gate, and $[P(y)+P(\omega y)]$ encodes the two inputs of that gate, which equals to $P(\omega^2 y)$ (where $\omega^2 y$ encodes the output of the circuit). At the same time, since the gate is addition, the right operand $((1 - S(y)) \cdot \ldots)$ must evaluated to zero. The same goes for when $S(y) = 0$, i.e., multiplication gate.

Overall, another zero-test on H_{gates}.

Wirings are encoded correctly.
For example, the input 6 flows to right input of Gate0 and left input of Gate1, we need to prove that does data flows (wiring) are encoded correctly. For our examples, the equivalent constraints are:

$$\begin{cases} P(\omega^{-2}) = P(\omega^1) = P(\omega^3) \\ P(\omega^{-1}) = P(\omega^0) \\ P(\omega^2) = P(\omega^6) \\ P(\omega^3) = P(\omega^4) \end{cases}$$

To do so, define a rotation polynomial $W : H \to H$ such that:

$$\begin{cases} W(\omega^{-2}, \omega^1, \omega^3) = (\omega^3, \omega^{-2}, \omega^1) \\ W(\omega^{-1}, \omega^0) = (\omega^0, \omega^1) \\ \dots \end{cases}$$

Lemma 4. $\forall y \in H : P(y) = P(W(y)) \Rightarrow$ *wiring constraints are satisfied.*

Since W has degree of d and P has degree of d, the verification can takes quadratic time. The trick here is to use prod-check (another IOP check) to reduce it to linear complexity. Not sure how to yet, another time. :P

Outputs are encoded correctly (is zero).
Just let prover to open P at the output of the final gate.

2 STARK