fflonk: cheaply opening many polynomials using the fast-fourier equation

Ariel Gabizon (based on work with Zac Williamson)

Motivation: Save gas on Ethereum

- Recent polynomial IOP based snarks (Sonic, Marlin, Plonk,..) verification consists of two pairings and some number $\mathbf{t}$ of $\mathbf{G}_1$ scalar multiplications.

Motivation: Save gas on Ethereum

- Recent polynomial IOP based snarks (Sonic, Marlin, Plonk,..) verification consists of two pairings and some number $\mathbf{t}$ of $\mathbf{G}_1$ scalar multiplications.
- Each $\mathbf{G}_1$ scalar mult - 6000 gas 5usd.
- That's 76800usd per scalar mult when doing an on chain proof once in half an our per scalar mult.

Motivation: Save gas on Ethereum

- Recent polynomial IOP based snarks (Sonic, Marlin, Plonk,..) verification consists of two pairings and some number $\mathbf{t}$ of $\mathbf{G}_1$ scalar multiplications.
- Each $\mathbf{G}_1$ scalar mult - 6000 gas 5usd.
- That's 76800usd per scalar mult when doing an on chain proof once in half an our per scalar mult.
- **This work:** Getting $\mathbf{t}$ down from 16 to 5 in plonk (at the cost of trippling prover time) .

snark verification reduces to polynomial commitment scheme (PCS) opening verification -

a 3 minute reminder on the Kate-Zaverucha-Goldberg PCS

Polynomial commitment schemes [KZG, 10]

- Prover send short commitment $\mathrm{cm}(\mathbf{f})$ to polynomial.

Polynomial commitment schemes [KZG, 10]

- Prover send short commitment $\mathrm{cm}(\mathbf{f})$ to polynomial.
- Later Verifier can choose value $s \in \mathbb{F}$.

Polynomial commitment schemes [KZG, 10]

- Prover send short commitment $\mathrm{cm}(\mathbf{f})$ to polynomial.
- Later Verifier can choose value $s \in \mathbb{F}$.
- Prover sends back $z = \mathbf{f}(s)$; together with proof $\mathrm{open}(\mathbf{f}, s)$ that z is correct.

Polynomial commitment schemes [KZG, 10]

- Prover send short commitment $\mathrm{cm}(\mathbf{f})$ to polynomial.
- Later Verifier can choose value $\mathbf{s} \in \mathbb{F}$.
- Prover sends back $\mathbf{z} = \mathbf{f}(\mathbf{s})$; together with proof $\mathrm{open}(\mathbf{f}, \mathbf{s})$ that $\mathbf{z}$ is correct.

KZG give us PCS with commitments and openings are practically 32 bytes.
Notation: $[\mathbf{x}]_1 = \mathbf{x} \cdot \mathbf{g}$ where $\mathbf{g}$ generator of (first source group of) elliptic curve group with pairing.

Setup: $[1]_1, [x]_1, \ldots, \left[x^{d}\right]_1$, for random $x \in \mathbb{F}$.

Setup: $[1]_1, [x]_1, \ldots, \left[x^{d}\right]_1$, for random $x \in \mathbb{F}$.

$\mathrm{cm}(f) := [f(x)]_1$

Setup: $[1]_1, [x]_1, \ldots, \left[x^d\right]_1$, for random $x \in \mathbb{F}$.

$\mathsf{cm}(f) := [f(x)]_1$

$\mathsf{open}(f, s) := [h(x)]_1$, where $h(X) := \frac{f(X) - f(s)}{X - s}$

Setup: $[1]_1, [x]_1, \ldots, \left[x^d\right]_1$, for random $x \in \mathbb{F}$.

$\mathrm{cm}(f) := [f(x)]_1$

$\mathrm{open}(f, s) := [h(x)]_1$, where $h(X) := \frac{f(X) - f(s)}{X - s}$

$\mathrm{verify}(\mathrm{cm}, \pi, z, s):$

$$e(\mathrm{cm} - [z]_1, [1]_1) \stackrel{?}{=} e(\pi, [x - s]_1)$$

Opening many polynomials at s

Input: $\mathbf{f}_0, \ldots \mathbf{f}_{\mathbf{d}-1}$, $z_0 = \mathbf{f}_0(\mathbf{s}), \ldots, z_{\mathbf{d}-1} = \mathbf{f}_{\mathbf{d}-1}(\mathbf{s})$.
Verifier has commitments $\mathsf{cm}_{\mathbf{i}}$ to $\mathbf{f}_{\mathbf{i}}$'s wants to verifier correctness of z's.

Opening many polynomials at s

Input: $f_0, \ldots f_{d-1}$, $z_0 = f_0(s), \ldots, z_{d-1} = f_{d-1}(s)$.
Verifier has commitments cm_i to f_i's wants to verifier correctness of z's.

Naive solution: Run KZG for each f_i.
Cost: d group elements in proof, d pairings for verifier

Batched opening (Sonic,Marlin,Plonk)

- Verifier sends random $\boldsymbol{\gamma} \in \mathbb{F}$

Batched opening (Sonic,Marlin,Plonk)

- Verifier sends random $\gamma \in \mathbb{F}$
- Prover computes combination $f(X) := \sum_{i<d} \gamma^i f_i(X)$
- Verifier computes commitment to f as $cm(f) := \sum_{i<d} \gamma^i cm_i$
- Prover and verifier use KZG to verify $f(s) = z$ for $z = \sum_{i<d} \gamma^i z_i$

cost: $d - 1$ verifier scalar muls to compute $cm(f)$
Punchline of this work: can get rid of this d dependence when s is a d'th power

A useful tool from BDFG/Halo infinite

thm[BDFG]: We can open a polynomial $\mathbf{f}$ at points $\mathbf{s}_0, \ldots, \mathbf{s}_{\mathbf{d}-1}$ with 2 verifier scalar mults no matter how large $\mathbf{d}$ is.

A useful tool from BDFG/Halo infinite

thm[BDFG]: We can open a polynomial $\mathbf{f}$ at points $\mathbf{s}_0, \ldots, \mathbf{s}_{\mathbf{d}-1}$ with 2 verifier scalar mults no matter how large $\mathbf{d}$ is.

*If we can reduce opening many polys at $\mathbf{s}$ to opening *one poly* at many points, we can use BDFG to get our desired result*

fflonk central idea: reduce opening many polys to one using the FFT identity

polys f_0, f_1, $a = f_0(s), b = f_1(s)$

fflonk central idea: reduce opening many polys to one using the FFT identity

polys f_0, f_1, $a = f_0(s), b = f_1(s)$

First attempt: Only open the sum - $F(X) := f_0(X) + f_1(X)$. Prove that $F(s) = c = a + b$.

fflonk central idea: reduce opening many polys to one using the FFT identity

polys f_0, f_1, $a = f_0(s), b = f_1(s)$

First attempt: Only open the sum - $F(X) := f_0(X) + f_1(X)$. Prove that $F(s) = c = a + b$.
problem: Doesn't constrain a, b individually - for any a', $(a', c - a')$ will also verify

fflonk central idea: reduce opening many polys to one using the FFT identity

polys f_0, f_1, $a = f_0(s), b = f_1(s)$

First attempt: Only open the sum - $F(X) := f_0(X) + f_1(X)$. Prove that $F(s) = c = a + b$.
problem: Doesn't constrain a, b individually - for any a', $(a', c - a')$ will also verify

Solution: Use "FFT equation in reverse direction":

$$\mathbf{F}(\mathbf{X}) = \mathbf{f}_0(\mathbf{X}^2) + \mathbf{X}\mathbf{f}_1(\mathbf{X}^2)$$

To commit to $\mathbf{f}_0, \mathbf{f}_1$ send $\mathsf{cm}(\mathbf{F})$.

Solution: Use "FFT equation in reverse direction":

$$F(X) = f_0(X^2) + Xf_1(X^2)$$

To commit to f_0, f_1 send $\mathrm{cm}(F)$.

Assume $s = t^2$.
To open $f_0(s), f_1(s)$ open F at $\{t, -t\}$:

Solution: Use "FFT equation in reverse direction":

$$F(X) = f_0(X^2) + X f_1(X^2)$$

To commit to f_0, f_1 send $\mathrm{cm}(F)$.

Assume $s = t^2$.
To open $f_0(s), f_1(s)$ open F at $\{t, -t\}$:

$$b_0 := F(t) = f_0(s) + t f_1(s) = a + tb$$

$$b_1 := F(-t) = f_0(s) - t f_1(s) = a - tb$$

i.e. (b_0, b_1) give two independent constraints on (a, b)!

Assume $s = t^2$.
To open $f_0(s), f_1(s)$ open F at $\{t, -t\}$:

Assume $s = t^2$.
To open $f_0(s), f_1(s)$ open F at $\{t, -t\}$:

$$b_0 := F(t) = f_0(s) + t f_1(s) = a + tb$$

$$b_1 := F(-t) = f_0(s) - t f_1(s) = a - tb$$

i.e. (b_0, b_1) give two independent constraints on (a, b)!

- Similar construction can open d polys at any $s = t^d$
- *Important:* poly-iop based snarks work fine with a PCS that can only open d'th powers.

Cheaply opening a poly at d points [BDFG]

Given poly f with commitment cm, $s_0, \dots, s_{d-1} \in \mathbb{F}$, suppose $z_i = f(s_i)$ for $i < d$.

- Define poly r of degree $< d$ with $r(s_i) = z_i$ for $i < d$.

Cheaply opening a poly at d points [BDFG]

Given poly f with commitment cm, $s_0, \ldots, s_{d-1} \in \mathbb{F}$, suppose $z_i = f(s_i)$ for $i < d$.

- Define poly r of degree $< d$ with $r(s_i) = z_i$ for $i < d$.
- Define $Z(X) = \prod_{i<d}(X - s_i)$. Then we have $Z|(f - r)$.

Cheaply opening a poly at d points [BDFG]

Given poly f with commitment cm, $s_0, \ldots, s_{d-1} \in \mathbb{F}$, suppose $z_i = f(s_i)$ for $i < d$.

- Define poly r of degree $< d$ with $r(s_i) = z_i$ for $i < d$.
- Define $Z(X) = \prod_{i<d}(X - s_i)$. Then we have $Z|(f - r)$.
- Let $h(X) := (f(X) - r(X))/(Z(X)$. Prover sends $\pi = [h(x)]_1$.

Cheaply opening a poly at d points [BDFG]

Given poly f with commitment cm, $s_0, \ldots, s_{d-1} \in \mathbb{F}$, suppose $z_i = f(s_i)$ for $i < d$.

- Define poly r of degree $< d$ with $r(s_i) = z_i$ for $i < d$.
- Define $Z(X) = \prod_{i<d}(X - s_i)$. Then we have $Z|(f - r)$.
- Let $h(X) := (f(X) - r(X))/(Z(X)$. Prover sends $\pi = [h(x)]_1$.
- From [KZG]: Verifier can now check

$$e(\mathrm{cm} - [r(x)]_1, 1) = e(\pi, [Z(x)]_2$$

Cheaply opening a poly at $\mathbf{d}$ points [BDFG]

Given poly $\mathbf{f}$ with commitment cm, $\mathbf{s}_0, \ldots, \mathbf{s}_{\mathbf{d}-1} \in \mathbb{F}$, suppose $\mathbf{z}_\mathbf{i} = \mathbf{f}(\mathbf{s}_\mathbf{i})$ for $\mathbf{i} < \mathbf{d}$.

- Define poly $\mathbf{r}$ of degree $< \mathbf{d}$ with $\mathbf{r}(\mathbf{s}_\mathbf{i}) = \mathbf{z}_\mathbf{i}$ for $\mathbf{i} < \mathbf{d}$.
- Define $\mathbf{Z}(\mathbf{X}) = \prod_{\mathbf{i}<\mathbf{d}}(\mathbf{X} - \mathbf{s}_\mathbf{i})$. Then we have $\mathbf{Z} | (\mathbf{f} - \mathbf{r})$.
- Let $\mathbf{h}(\mathbf{X}) := (\mathbf{f}(\mathbf{X}) - \mathbf{r}(\mathbf{X}))/(\mathbf{Z}(\mathbf{X})$. Prover sends $\boldsymbol{\pi} = [\mathbf{h}(\mathbf{x})]_1$.
- From [KZG]: Verifier can now check

$$\mathbf{e}(\mathrm{cm} - [\mathbf{r}(\mathbf{x})]_1, 1) = \mathbf{e}(\boldsymbol{\pi}, [\mathbf{Z}(\mathbf{x})]_2$$

This requires $\mathbf{d}$ G1 and G2 verifier scalar muls!

In [BDFG] we trade these scalar mults for an extra group element in the proof:

- Verifier chooses random $\alpha \in \mathbb{F}$ and sends to prover.
- Define the polynomial

$$\mathbf{L}(\mathbf{X}) := \mathbf{f}(\mathbf{X}) - \mathbf{r}(\alpha) - \mathbf{Z}(\alpha)\mathbf{h}(\mathbf{X})$$

In [BDFG] we trade these scalar mults for an extra group element in the proof:

- Verifier chooses random $\alpha \in \mathbb{F}$ and sends to prover.
- Define the polynomial

$$\mathbf{L}(\mathbf{X}) := \mathbf{f}(\mathbf{X}) - \mathbf{r}(\alpha) - \mathbf{Z}(\alpha)\mathbf{h}(\mathbf{X})$$

- If evals are correct, $\mathbf{L}(\mathbf{X})$ should be zero at α

In [BDFG] we trade these scalar mults for an extra group element in the proof:

- Verifier chooses random $\alpha \in \mathbb{F}$ and sends to prover.
- Define the polynomial

$$\mathbf{L}(\mathbf{X}) := \mathbf{f}(\mathbf{X}) - \mathbf{r}(\alpha) - \mathbf{Z}(\alpha)\mathbf{h}(\mathbf{X})$$

- If evals are correct, $\mathbf{L}(\mathbf{X})$ should be zero at α
- Verifer can compute $\mathsf{cm}(\mathbf{L})$ with only two scalar muls:

$$\mathsf{cm}(\mathbf{L}) = \mathsf{cm} - [\mathbf{r}(\alpha)]_1 - \mathbf{Z}(\alpha)\pi$$

- Prover and verifier can now use KZG to check $\mathbf{L}(\alpha) = 0$.