fflonK for the Polygon zkEVM

Héctor Masip Ardevol

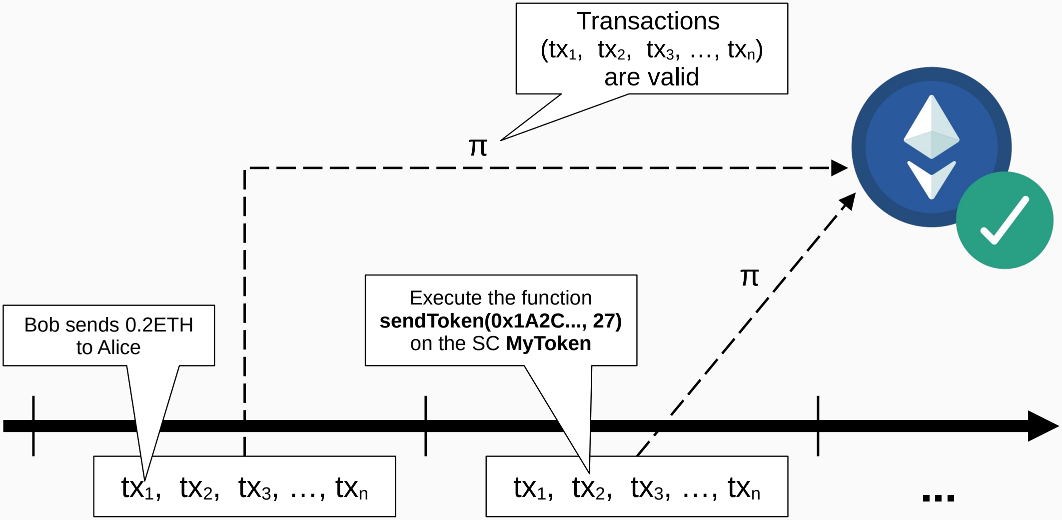
Joint work with **Polygon zkEVM**

Information Security Group (ISG),
Universitat Politècnica de Catalunya (UPC)

April 4th 2023 - Lisbon

Table of Contents

Which is the Finality of a zkEVM?



Statistics of the Polygon zkEVM Circuit

Some interesting numbers for the circuit C attesting the validity of a batch (≈ 500 standard) of transactions:

a) **Polynomials**:

1. Total number of polynomials: **1276**.
2. Number of witness polynomials: **1058**.
3. Number of preprocessed polynomials: **218**.
4. Degree's bound of polynomials: $n = 2^{23}$.

b) **Constraints**:

5. Number of AIR constraints: **631** (with degree's bound of $3n$).
6. Number of inclusion constraints: **28**.
7. Number of connection constraints: **2**.
8. Number of multiset equality constraints: **18**.

Working over the prime field $\mathbb{F}_p$ with $p = 2^{64} - 2^{32} + 1$, this means that:

The (non-encoded) execution trace is around **86GB**.

Statistics of the Polygon zkEVM Circuit

Some interesting numbers for the circuit C attesting the validity of a batch (≈ 500 standard) of transactions:

a) **Polynomials**:

1. Total number of polynomials: **1276**.
2. Number of witness polynomials: **1058**.
3. Number of preprocessed polynomials: **218**.
4. Degree's bound of polynomials: $n = 2^{23}$.

b) **Constraints**:

5. Number of AIR constraints: **631** (with degree's bound of $3n$).
6. Number of inclusion constraints: **28**.
7. Number of connection constraints: **2**.
8. Number of multiset equality constraints: **18**.

Working over the prime field $\mathbb{F}_p$ with $p = 2^{64} - 2^{32} + 1$, this means that:

The (non-encoded) execution trace is around **86GB**.

Statistics of the Polygon zkEVM Circuit

Some interesting numbers for the circuit C attesting the validity of a batch (≈ 500 standard) of transactions:

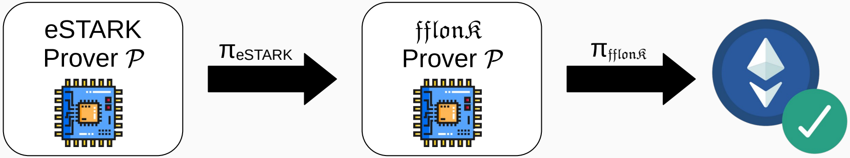
a) **Polynomials**:
   1. Total number of polynomials: **1276**.
   2. Number of witness polynomials: **1058**.
   3. Number of preprocessed polynomials: **218**.
   4. Degree's bound of polynomials: $n = 2^{23}$.

b) **Constraints**:
   5. Number of AIR constraints: **631** (with degree's bound of $3n$).
   6. Number of inclusion constraints: **28**.
   7. Number of connection constraints: **2**.
   8. Number of multiset equality constraints: **18**.

Working over the prime field $\mathbb{F}_p$ with $p = 2^{64} - 2^{32} + 1$, this means that:

The (non-encoded) execution trace is around **86GB**.

- To generate a SNARK for this gigantic circuit C we need a very **fast prover**.
- Since the proof will be verified on-chain, we have also required a **small proof size** and a **fast verifier**.
- **Solution:** Compose a SNARK $\mathcal{I}$ that features a fast prover with another SNARK $\mathcal{O}$ that boasts a small proof size and a fast verifier.
- We chose eSTARK[1] (very fast prover, but long proof size) for $\mathcal{I}$ and $\mathfrak{fflonk}$ (slow prover, but constant proof size and verification time) for $\mathcal{O}$.



[1] This proving system is precisely the STARK proving system with support for arguments.

SNARKs with Constant Proof Size and Verification Time

Scheme	Universal TS	CRS/SRS Size	Proving Time	Proof Size	Ver. Time
Groth16	✗	$3m + w\ \mathbb{G}_1$	$3m + w - \ell\ \mathbb{G}_1, m\ \mathbb{G}_2$	$2\ \mathbb{G}_1, 1\ \mathbb{G}_2$	$\ell\ \mathbb{G}_1, 3\ \mathsf{P}$
$\mathscr{P}$lon$\mathscr{K}$	✓	$3n\ \mathbb{G}_1, 2\ \mathbb{G}_2$	$11n\ \mathbb{G}_1$	$7\ \mathbb{G}_1, 7\ \mathbb{F}$	$16\ \mathbb{G}_1, 2\ \mathsf{P}$
$\mathscr{f}\mathscr{f}$lon$\mathscr{K}$	✓	$9n\ \mathbb{G}_1, 2\ \mathbb{G}_2$	$35n\ \mathbb{G}_1$	$4\ \mathbb{G}_1, 15\ \mathbb{F}$	$5\ \mathbb{G}_1, 2\ \mathsf{P}$

- m denotes the number of multiplication gates.
- w denotes the number of wires.
- n denotes the number of gates.
- ℓ denotes the number of public inputs ($\ell = 1$ in our case).
- $\mathbb{G}_i$ denotes scalar multiplications in $\mathbb{G}_i$.
- P denotes pairings.

SNARKs with Constant Proof Size and Verification Time

Scheme	Universal TS	CRS/SRS Size	Proving Time	Proof Size	Ver. Time
Groth16	✗	$3m + w\ \mathbb{G}_1$	$3m + w - \ell\ \mathbb{G}_1, m\ \mathbb{G}_2$	$2\ \mathbb{G}_1, 1\ \mathbb{G}_2$	≈ 232.000 gas
$\mathcal{P}$lon$\mathcal{K}$	✓	$3n\ \mathbb{G}_1, 2\ \mathbb{G}_2$	$11n\ \mathbb{G}_1$	$7\ \mathbb{G}_1, 7\ \mathbb{F}$	≈ 285.000 gas
$\mathcal{ff}$lon$\mathcal{K}$	✓	$9n\ \mathbb{G}_1, 2\ \mathbb{G}_2$	$35n\ \mathbb{G}_1$	$4\ \mathbb{G}_1, 15\ \mathbb{F}$	≈ 185.000 gas

- m denotes the number of multiplication gates.
- w denotes the number of wires.
- n denotes the number of gates.
- ℓ denotes the number of public inputs ($\ell = 1$ in our case).
- $\mathbb{G}_i$ denotes scalar multiplications in $\mathbb{G}_i$.
- **P** denotes pairings.

Proof System Diagram

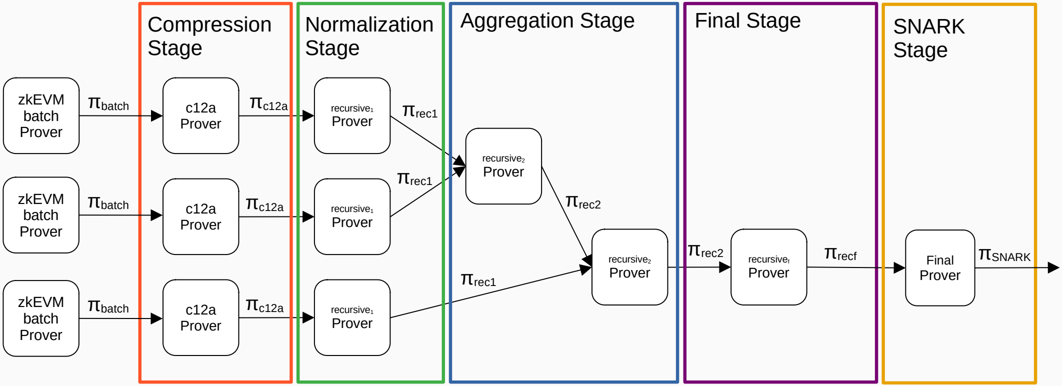


Table of Contents

What is a Polynomial Commitment Scheme (PCS)? i

Given the polynomial family $\mathcal{F} = \mathbb{F}_{<d}[X]$ of polynomials of degree lower than d with coefficients over a finite field $\mathbb{F}$, a PCS works as follows:

The Prover $\mathcal{P}(\mathcal{F}, f)$		The Verifier $\mathcal{V}(\mathcal{F})$
commit$(f) = \mathrm{com}_f$		
	$\xrightarrow{\mathrm{com}_f}$	
		Samples an evaluation challenge $x \in \mathbb{F}$
	$\xleftarrow{x}$	
Computes proof π that: $f(x) = y$		
	$\xrightarrow{y, \pi}$	
		Accepts or rejects

What is a Polynomial Commitment Scheme (PCS)? ii

Definition 1

A **Polynomial Commitment Scheme (PCS)** is a tuple (***setup***, ***commit***, ***open***) such that:

- ***setup***$(1^\lambda) = \mathrm{gp}$. Outputs the public **group parameters** gp.
- ***commit***$(\mathrm{gp}, f, r) = \mathrm{com}_f$. Outputs a commitment to $f \in \mathcal{F}$ with[2] $r \in \mathbb{F}$.
- ***open***(gp, f, x, y) is a (public coin) protocol between $\mathcal{P}$ and $\mathcal{V}$ such that:
  1. $\mathcal{P}(\mathrm{gp}, f, x, y) = \pi$.
  2. $\mathcal{V}(\mathrm{gp}, \mathrm{com}_f, x, y, \pi) = \mathrm{accept/reject}$.

In ***open*** is turned non-interactive, then a PCS is a (zk-)SNARK for the statement:

"I know an $f \in \mathcal{F}$ such that $f(x) = y$ and ***commit***$(\mathrm{gp}, f, r) = \mathrm{com}_f$."

[2] The commitment scheme is statistically binding and computationally hiding, but r can be used to make it computationally binding and statistically hiding.

Example of PCS: KZG

setup(1^λ): The setup algorithm works by sampling a random $s \in \mathbb{F}$, computing $\mathrm{gp} = ([1]_1, [s]_1, \ldots, [s^{d-1}]_1, [1]_2, [s]_2)$ and **deleting** s.

The Prover $\mathcal{P}(\mathcal{F}, f)$		The Verifier $\mathcal{V}(\mathcal{F})$
commit$(\mathrm{gp}, f, r) = f(s)\mathbb{G}_1 := [f(s)]_1$	$[f(s)]_1$ $\longrightarrow$	
		Samples an evaluation challenge $x \in \mathbb{F}$
	x $\longleftarrow$	
Computes $f(x) = y$, the polynomial $q(X) = \frac{f(X) - y}{X - x}$, and the proof $\pi := [q(s)]_1$.		
	$y, [q(s)]_1$ $\longrightarrow$	
		Outputs accepts if: $e([q(s)]_1, [s]_2 - [x]_2) = e([f(s)]_1 - [y]_1, [1]_2)$, rejecting otherwise.

Properties of KZG

- The algorithm ***setup***(1^λ) requires to be trusted on deleting s.
- $\mathcal{P}$ runs in $O(d)$ since it computes:
  a) The MSM $f(s)\mathbb{G}_1 = f_0 \cdot [1]_1 + f_1 \cdot [s]_1 + \cdots + f_{d-1} \cdot [s^{d-1}]_1$.
  b) The division $q(X) = (f(X) - y)/(X - x)$.
  c) The MSM $q(s)\mathbb{G}_1 = q_0 \cdot [1]_1 + q_1 \cdot [s]_1 + \cdots + q_{d-2} \cdot [s^{d-2}]_1$.
- The proof π consists of a single $\mathbb{G}_1$-element.
- $\mathcal{V}$ runs in $O(1)$ time since it computes 2 pairings.
- It can be made zero-knowledge by masking f with r [ZGK+18].
- Direct generalizations: **batch openings, multiple polynomials and both**.

Simple KZG Generalizations

Batch Openings: Open f at $x_1, \ldots, x_m$.

- Compute the polynomial $r \in \mathbb{F}_{<m}[X]$ s.t. $r(x_j) = f(x_j)$.
- Compute the quotient $q(X) = (f(X) - r(X)) / \prod_{j=1}^{m}(X - x_j)$.
- Verifying q is a polynomial implies $f(x_j) = y_j$, for $j \in [m]$.

Multiple Polynomials: Open $f_1, \ldots, f_n$ at x.

- Compute the quotient $q_i(X) = (f_i(X) - y_i)/(X - x)$ for each $i \in [n]$.
- Mix all the resulting quotients with a random linear combination q.
- Verifying q is a polynomial implies each q_i is a polynomial, for $i \in [n]$.

Simple KZG Generalizations

Batch Openings: Open f at $x_1, \ldots, x_m$.

- Compute the polynomial $r \in \mathbb{F}_{<m}[X]$ s.t. $r(x_j) = f(x_j)$.
- Compute the quotient $q(X) = (f(X) - r(X))/\prod_{j=1}^{m}(X - x_j)$.
- Verifying q is a polynomial implies $f(x_j) = y_j$, for $j \in [m]$.

Multiple Polynomials: Open $f_1, \ldots, f_n$ at x.

- Compute the quotient $q_i(X) = (f_i(X) - y_i)/(X - x)$ for each $i \in [n]$.
- Mix all the resulting quotients with a random linear combination q.
- Verifying q is a polynomial implies each q_i is a polynomial, for $i \in [n]$.

Generalization: Batch Openings and Multiple Polynomials ($\mathscr{P}\mathfrak{lon}\mathscr{K}$ Version) (*)

The Prover $\mathcal{P}(\mathcal{F}, f_1, \ldots, f_{n_1}, f'_1, \ldots, f'_{n_2})$		The Verifier $\mathcal{V}(\mathcal{F})$
commit$(\text{gp}, f_i, r_i) = [f_i(s)]_1$, for $i \in [n_1]$		
commit$(\text{gp}, f'_i, r'_i) = \left[f'_i(s)\right]_1$, for $i \in [n_2]$		
	$[f_1(s)]_1, \ldots, [f_{n_1}(s)]_1$ $\longrightarrow$ $[f'_1(s)]_1, \ldots, [f'_{n_2}(s)]_1$	
		Samples evaluation challenges $x, x' \in \mathbb{F}$
	$\longleftarrow$ x, x'	
Computes $f_i(x) = y_i$ and $f'_i(x') = y'_i$ the polynomials: $q(X) = \sum_{i=1}^{n_1} \alpha^{i-1} \cdot \frac{f_i(X) - y_i}{X - x}$ $q'(X) = \sum_{i=1}^{n_2} (\alpha')^{i-1} \cdot \frac{f'_i(X) - y'_i}{X - x'}$ and the proofs $\pi = [q(s)]_1$, $\pi' = \left[q'(s)\right]_1$		
	$y_1, \ldots, y_{n_1}, [q(s)]_1$ $\longrightarrow$ $y'_1, \ldots, y'_{n_2}, \left[q'(s)\right]_1$	Computes $[F(s)]_1 = \sum_{i=1}^{n_1} \alpha^{i-1} \cdot [f_i(s)]_1 + r \cdot \sum_{i=1}^{n_2} (\alpha')^{i-1} \cdot [f'_i(s)]_1$ and $Y = \sum_{i=1}^{n_1} \alpha^{i-1} \cdot y_i + r \cdot \sum_{i=1}^{n_2} (\alpha')^{i-1} \cdot y'_i$
		Outputs accepts if: $e\left([q(s)]_1 + r \cdot \left[q'(s)\right]'_1, [s]_2\right) = e\left([F(s)]_1 - [Y]_1 + x \cdot [q(s)]_1 + r \cdot x \cdot \left[q'(s)\right]_1, [1]_2\right)$ rejecting otherwise.

$\mathscr{P}\mathfrak{lon}\mathscr{K}$ KZG: Complexity (*)

Theorem 2 (Worst case KZG Complexity)

Let $f_1, \dots, f_{n_1}, f'_1, \dots, f'_{n_2} \in \mathcal{F}$ be of degree $d-1$ such that any of them have zero coefficients. The (n_1, n_2)-pols and $(1,1)$-openings version of the KZG polynomial commitment scheme has the following measures:

1. ***Proving Time**: $(n_1 + n_2 + 2)d - 2$ escalar multiplications over $\mathbb{G}_1$.*
2. ***Proof Size**: $(n_1 + n_2 + 2)$ $\mathbb{G}_1$-elements and $(n_1 + n_2 + 2)$ $\mathbb{F}$-elements.*
3. ***Verification Time**: $(n_1 + n_2 + 2)$ escalar multiplications over $\mathbb{G}_1$ and 2 pairings.*

- **Problem**: The verification complexity is dominated by the scalar multiplications performed over the $\mathbb{G}_1$-elements in the proof.
- **Solution**: Reduce $\mathbb{G}_1$-elements in the proof.

- **Claim 1**: In general, $f(x_i) = y_i$ for $i \in [n]$ if and only if $q(X) = (f(X) - r(X))/Z_S(X)$ is a polynomial of degree $\deg(f) - |S|$.
- **Claim 2**: Even more general, $f(x_i) = y_i$ and $f'(x'_i) = y'_i$ for $i \in [n]$ if and only if $q(X) = (f(X) - r(X))/Z_S(X) + \alpha \cdot (f'(X) - r'(X))/Z_{S'}(X)$ is a polynomial of degree $d = \max\{\deg(f) - |S|, \deg(f') - |S'|\}$, where $\alpha \in \mathbb{F}$ is a uniformly sampled value.

Lemma 3 ($\mathfrak{shplonK}$ [BDFG20])

$f(x_i) = y_i$ and $f'(x'_i) = y'_i$ for $i \in [n]$ if and only if the following polynomial is of degree d:

$$L(X) = \frac{Z_{S'}(y)(f(X) - r(y)) + \alpha \cdot Z_S(y)(f'(X) - r'(y)) - Z_{SS'}(y) \cdot q(X)}{X - y}$$

***Put simply**: Validating the $|S| + |S'|$ openings of q is equivalent to validating the opening at 0 of L (i.e., **the verifier complexity does not grow with the number of openings**).*

- The **combine** $C\colon \mathbb{F}_{<d}[X]^t \to \mathbb{F}_{<dt}[X]$ function is defined as follows:

$$C(f_1, \ldots, f_t) := \sum_{i=1}^{t} f_i(X^t) \cdot X^{i-1}.$$

Lemma 4 ($\mathfrak{c}$-$\mathfrak{shplonK}$ [GW21])

Opening $f_1, \ldots, f_t \in \mathbb{F}[X]$ at $x \in \mathbb{F}$ is the equivalent to opening C at the t-roots of x, that is, the t solutions of:

$$z^t = x \pmod{p}. \tag{1}$$

In fact, if $z \in \mathbb{F}$ is a solution of (1), then so are $z \cdot \omega_t^i$, for $i \in [t]$.

The Prover $\mathcal{P}(\mathcal{F}, f_1, \ldots, f_{n_1}, f'_1, \ldots, f'_{n_2})$		The Verifier $\mathcal{V}(\mathcal{F})$
Compute the polynomial **combinations:** $$C(X) = \sum_{i=1}^{n_1} f_i(X^{n_1}) \cdot X^{i-1}$$ $$C'(X) = \sum_{i=1}^{n_2} f'_i(X^{n_2}) \cdot X^{i-1}$$ and their commitment $[C(s)]_1$, $[C'(s)]_1$		
	$[C(s)]_1, [C'(s)]_1$ $\longrightarrow$	
		Samples evaluation challenges $x, x' \in \mathbb{F}$
	x, x' $\longleftarrow$	
Computes $f_i(x) = y_i$ and $f'_i(x') = y'_i$		
	$y_1, \ldots, y_{n_1}, y'_1, \ldots, y'_{n_2}$ $\longrightarrow$	

A Verifier-Friendly PCS: $\mathfrak{c\text{-}shplonk}$ ii

The Prover $\mathcal{P}(\mathcal{F}, f_1, \ldots, f_{n_1}, f'_1, \ldots, f'_{n_2})$		The Verifier $\mathcal{V}(\mathcal{F})$
	$y_1, \ldots, y_{n_1}, y'_1, \ldots, y'_{n_2}$ $\longrightarrow$	
.... Batching Round		
	$\longleftarrow$ α	Samples a batching challenge $\alpha \in \mathbb{F}$
Compute the following:		
a) The n_1-roots of x $S = \{z_1, \ldots, z_{n_1}\}$ and the n_2-roots of x' $S' = \{z'_1, \ldots, z'_{n_2}\}$		
b) The polynomials $r \in \mathbb{F}_{<\|S\|}[X]$ s.t. $C(z_j) = r(z_j)$ and $r' \in \mathbb{F}_{<\|S'\|}[X]$ s.t. $C'(z'_j) = r'(z'_j)$		
c) The zerofiers $Z_S(X) = \prod_{z \in S}(X - z)$ and $Z_{S'}(X) = \prod_{z \in S'}(X - z)$		
Compute the batching polynomial: $q(X) = \frac{C(X) - r(X)}{Z_S(X)} + \alpha \frac{C'(X) - r'(X)}{Z_{S'}(X)}$		
and its commitment $[q(s)]_1$	$[q(s)]_1$ $\longrightarrow$	

A Verifier-Friendly PCS: $\mathfrak{c\text{-}shplonK}$ iii

The Prover $\mathcal{P}(\mathcal{F}, f_1, \ldots, f_{n_1}, f'_1, \ldots, f'_{n_2})$		The Verifier $\mathcal{V}(\mathcal{F})$
	$\xrightarrow{[q(s)]_1}$	
	Reduction Round	
	$\xleftarrow{y}$	Samples a reduction challenge $y \in \mathbb{F}$
Compute the reduced polynomial: $L(X) = Z_{S'}(y)(C(X) - r(y)) + \alpha \cdot Z_S(y)(C'(X) - r'(y)) - Z_{SS'}(y) \cdot q(X)$ and the commitment $[W]_1 := [L(s)/(s-y)]_1$	$\xrightarrow{[W]_1}$	
		Compute a), b) and c) as in $\mathcal{P}$'s batching round, obtaining the sets S, S' and the polynomials $r, r', Z_S, Z_{S'}$
		Computes $[F(s)]_1 = Z_{S'}(y)\,[C]_1 + \alpha \cdot Z_S(y)\left[C'\right]_1 - Z_{SS'}(y) \cdot [W]_1$ and $Y = r(y) + \alpha \cdot r'(y)$
		Outputs accepts if: $e\left([W]_1, [s]_2\right) = e\left([F(s)]_1 - [Y]_1 + y \cdot [W]_1, [1]_2\right)$ rejecting otherwise.

- Let $c_i = (n_i \cdot d + n_i - 1)$ .Assume w.l.o.g. that $n_1 > n_2$ and that $c_1 - |S| > c_2 - |S'|$.

Theorem 5 (Worst case $\mathfrak{c}$-$\mathfrak{shplonk}$ Complexity)

Let $f_1, \ldots, f_{n_1}, f'_1, \ldots, f'_{n_2} \in \mathcal{F}$ be of degree $d-1$ such that any of them have zero coefficients. The (n_1, n_2)-pols and $(1,1)$-openings version of the $\mathfrak{c}$-$\mathfrak{shplonk}$ polynomial commitment scheme has the following measures:

1. ***Proving Time**: $3c_1 + c_2 - 2|S|$ escalar multiplications over $\mathbb{G}_1$.*
2. ***Proof Size**: 4 $\mathbb{G}_1$-elements and $(n_1 + n_2 + 4)$ $\mathbb{F}$-elements.*
3. ***Verification Time**: 4 escalar multiplications over $\mathbb{G}_1$ and 2 pairings.*

Table of Contents

A Useful Observation

- The **combine** $C\colon \mathbb{F}_{<d}[X]^t \to \mathbb{F}_{<dt}[X]$ function is defined as follows:

$$C(f_1, \ldots, f_t) := \sum_{i=1}^{t} f_i(X^t) \cdot X^{i-1}.$$

- Let $f, g, h \in \mathbb{F}_{<4}[X]$. To obtain ***commit***(gp, C) observe that computing $f(X^t) \cdot X^i$ is a "multiply-index-by-t" (except for zero) followed by "shift-index-by-i":

$$\begin{array}{rl} f(X^3) & = [\; f_0, \; 0, \; 0, \; f_1, \; 0, \; 0, \; f_2, \; 0, \; 0, \; f_4, \; 0 \;\; 0 \;] \\ g(X^3) \cdot X & = [\; 0, \; g_0, \; 0, \; 0, \; g_1, \; 0, \; 0, \; g_2, \; 0, \; 0, \; g_4 \;\; 0 \;] \\ h(X^3) \cdot X^2 & = [\; 0, \; 0, \; h_0, \; 0, \; 0, \; h_1, \; 0, \; 0, \; h_2, \; 0, \; 0 \;\; h_4 \;] \end{array}$$

  and moreover:

  a) ***commit***$(C) =$ ***commit***$(f(X^3)) +$ ***commit***$(g(X^3)X) +$ ***commit***$(h(X^3)X^2)$ takes 15 scalar multiplications and 2 additions.

  b) ***commit***$(C) =$ ***commit***$(f(X^3) + g(X^3)X + h(X^3)X^2)$ takes 15 scalar multiplications.

  c) ***commit***(C) takes exactly dt $\mathbb{G}_1$ scalar multiplications.

A Useful Observation

- The **combine** $C\colon \mathbb{F}_{<d}[X]^t \to \mathbb{F}_{<dt}[X]$ function is defined as follows:

$$C(f_1, \ldots, f_t) := \sum_{i=1}^{t} f_i(X^t) \cdot X^{i-1}.$$

- Let $f, g, h \in \mathbb{F}_{<4}[X]$. To obtain ***commit***(gp, C) observe that computing $f(X^t) \cdot X^i$ is a "multiply-index-by-t" (except for zero) followed by "shift-index-by-i":

$$\begin{array}{rl} f(X^3) & = [\; f_0, \; 0, \; 0, \; f_1, \; 0, \; 0, \; f_2, \; 0, \; 0, \; f_4, \; 0 \; 0 \;] \\ g(X^3) \cdot X & = [\; 0, \; g_0, \; 0, \; 0, \; g_1, \; 0, \; 0, \; g_2, \; 0, \; 0, \; g_4 \; 0 \;] \\ h(X^3) \cdot X^2 & = [\; 0, \; 0, \; h_0, \; 0, \; 0, \; h_1, \; 0, \; 0, \; h_2, \; 0, \; 0 \; h_4 \;] \end{array}$$

and moreover:

a) ***commit***(C) = ***commit***$(f(X^3))$ + ***commit***$(g(X^3)X)$ + ***commit***$(h(X^3)X^2)$ takes 15 scalar multiplications and 2 additions.

b) ***commit***(C) = ***commit***$(f(X^3) + g(X^3)X + h(X^3)X^2)$ takes 15 scalar multiplications.

c) ***commit***(C) takes exactly dt $\mathbb{G}_1$ scalar multiplications.

A Useful Observation

- The **combine** $C\colon \mathbb{F}_{<d}[X]^t \to \mathbb{F}_{<dt}[X]$ function is defined as follows:

$$C(f_1, \dots, f_t) := \sum_{i=1}^{t} f_i(X^t) \cdot X^{i-1}.$$

- Let $f, g, h \in \mathbb{F}_{<4}[X]$. To obtain ***commit***(gp, C) observe that computing $f(X^t) \cdot X^i$ is a "multiply-index-by-t" (except for zero) followed by "shift-index-by-i":

$$\begin{array}{lllllllllllllll} f(X^3) & = [& f_0, & 0, & 0, & f_1, & 0, & 0, & f_2, & 0, & 0, & f_4, & 0 & 0 &] \\ g(X^3) \cdot X & = [& 0, & g_0, & 0, & 0, & g_1, & 0, & 0, & g_2, & 0, & 0, & g_4 & 0 &] \\ h(X^3) \cdot X^2 & = [& 0, & 0, & h_0, & 0, & 0, & h_1, & 0, & 0, & h_2, & 0, & 0 & h_4 &] \end{array}$$

and moreover:

a) ***commit***(C) = ***commit***$(f(X^3))$ + ***commit***$(g(X^3)X)$ + ***commit***$(h(X^3)X^2)$ takes 15 scalar multiplications and 2 additions.

b) ***commit***(C) = ***commit***$(f(X^3) + g(X^3)X + h(X^3)X^2)$ takes 15 scalar multiplications.

c) ***commit***(C) takes exactly dt $\mathbb{G}_1$ scalar multiplications.

Verifier Field Inversions

- Say that the verifier needs to perform the inversion of $x_1, \ldots, x_n \in \mathbb{F}$.
- Using Montgomery batch inversion we can convert the n inversions to 1 (16.000 gas) inversion and $3 \cdot (n-1)$ multiplications.
- **Problem**: The verifier still needs to perform 1 inversion.
- **Solution**: Let the prover do it for you!

The Prover $\mathcal{P}(\mathbb{F})$		The Verifier $\mathcal{V}(\mathbb{F})$
		Samples the inversion challenges $x_1, \ldots, x_n \in \mathbb{F}$
	$\xleftarrow{x_1, \ldots, x_n}$	
Computes the inversion of: $x = x_1 \cdot \ldots \cdot x_n$ Denote it as $\tilde{x}$		
	$\xrightarrow{\tilde{x}}$	
		Compute x and check $x \cdot \tilde{x} = 1$ Run batch inversion on $x, \tilde{x}$

Verifier Field Inversions

- Say that the verifier needs to perform the inversion of $x_1, \ldots, x_n \in \mathbb{F}$.
- Using Montgomery batch inversion we can convert the n inversions to 1 (16.000 gas) inversion and $3 \cdot (n-1)$ multiplications.
- **Problem**: The verifier still needs to perform 1 inversion.
- **Solution**: Let the prover do it for you!

The Prover $\mathcal{P}(\mathbb{F})$		**The Verifier** $\mathcal{V}(\mathbb{F})$
		Samples the inversion challenges $x_1, \ldots, x_n \in \mathbb{F}$
	$\xleftarrow{x_1, \ldots, x_n}$	
Computes the inversion of: $x = x_1 \cdot \ldots \cdot x_n$ Denote it as $\tilde{x}$		
	$\xrightarrow{\tilde{x}}$	
		Compute x and check $x \cdot \tilde{x} = 1$
		Run batch inversion on $x, \tilde{x}$

- Let $T = S_0 \cup S_1 \cup S_2$ where:

$$S_0 = h_0\langle\omega_0\rangle, \quad S_1 = h_1\langle\omega_1\rangle, \quad S_2 = h_2\langle\omega_2\rangle \cup h_3\langle\omega_2\rangle$$

  where $h_0^{|S_0|} = h_1^{|S_1|} = h_2^{|S_2|/2} = \mathfrak{z}$, $h_3^{|S_2|/2} = \mathfrak{z}\omega$ and $\omega_0, \omega_1, \omega_2, \omega$ are primitive roots of unity.
- In Round 4, we should divide a polynomial $f \in \mathbb{F}[X]$ of degree $\geq |T|$ by the zerofier over T:

$$Z_T(X) := \prod_{z \in T}(X - z)$$

- Naive polynomial long division would take (unparallelizable) $O(|T|^2)$ time. **Let's do it better!**

We start by noticing that:

$$Z_T(X) = Z_{S_0}(X) \cdot Z_{S_1}(X) \cdot Z_{S_2}(X) = (X^{|S_0|} - \mathfrak{z}) \cdot (X^{|S_1|} - \mathfrak{z}) \cdot (X^{|S_2|} - \mathfrak{z}(1+\omega)X^{|S_2|/2} + \mathfrak{z}^2\omega).$$

Then, we (sequentially) proceed as follows:

1. Divide f by Z_{S_0} to obtain the polynomial q_0 such that $q_0(X) \cdot Z_{S_0}(X) = f(X)$.
2. Divide q_0 by Z_{S_1} to obtain the polynomial q_1 such that $q_1(X) \cdot Z_{S_1}(X) = q_0(X)$.
3. Split $Z_{S_2}(X) = (X^{|S_2|} - \mathfrak{z}(1+\omega)X^{|S_2|/2} + \mathfrak{z}^2\omega)$ as the multiplication of the two inner zerofiers $(X^{|S_2|/2} - \mathfrak{z})$ and $(X^{|S_2|/2} - \mathfrak{z}\omega)$.
   Then:
   a) Divide q_1 by $(X^{|S_2|/2} - \mathfrak{z})$ to obtain the polynomial q_2 s.t. $q_2(X) \cdot (X^{|S_2|/2} - \mathfrak{z}) = q_1(X)$.
   b) Divide q_2 by $(X^{|S_2|/2} - \mathfrak{z}\omega)$ to obtain the polynomial q_3 s.t. $q_3(X) \cdot (X^{|S_2|/2} - \mathfrak{z}\omega) = q_2(X)$.

   The polynomial q_3 satisfies $q_3(X) \cdot Z_T(X) = f(X)$.

Lemma 6

Given a polynomial $f(X) = f_d X^d + \cdots + f_1 X + f_0 \in \mathbb{F}[X]$ of degree $d \geq m$ and a field element β, the quotient of the division $f(X)/(X^m - \beta)$ is the polynomial:

$$
\begin{aligned}
q(X) := {} & \left[f_d \cdot X^{d-m} + f_{d-1} \cdot X^{(d-1)-m} + \cdots + f_{d-(m-1)} \cdot X^{(d-(m-1))-m}\right] + \\
& + \left[(f_{d-m} + f_d \cdot \beta) \cdot X^{(d-m)-m} + \cdots + (f_{d-(2m-1)} + f_{d-(m-1)} \cdot \beta) \cdot X^{(d-(2m-1))-m}\right] + \\
& + \left[(f_{d-2m} + f_{d-m} \cdot \beta + f_d \cdot \beta^2) \cdot X^{(d-2m)-m} + \dots \right. \\
& \left. + (f_{d-(3m-1)} + f_{d-(2m-1)} \cdot \beta + f_{d-(m-1)} \cdot \beta^2) \cdot X^{(d-(3m-1))-m}\right] + \dots
\end{aligned}
$$

Division by $X^m - \beta$ (*) ii

- In words, q is a polynomial with the m leading coefficients equal to the m leading coefficients of f; the following m coefficients are of the form $f_i + f_j \cdot \beta$, with $j - i = m$; the following m coefficients are of the form $f_i + f_j \cdot \beta + f_k \cdot \beta^2$, with $j - i = k - j = m$; and so on.
- For instance, if $f(X) = \sum_{i=0}^{10} f_i X^i$ and $m = 2$, then:

$$\begin{aligned} q(X) := f_{10}X^8 + f_9X^7 + (f_8 + f_{10}\beta)X^6 + (f_7 + f_9\beta)X^5 + \\ + (f_6 + f_8\beta + f_{10}\beta^2)X^4 + (f_5 + f_7\beta + f_9\beta^2)X^3 + \\ + (f_4 + f_6\beta + f_8\beta^2 + f_{10}\beta^3)X^2 + (f_3 + f_5\beta + f_7\beta^2 + f_9\beta^3)X + \\ + (f_2 + f_4\beta + f_6\beta^2 + f_8\beta^3 + f_{10}\beta^4) \end{aligned}$$

- This division is 100% parallelizable.

Adding Zero-Knowledge (with Dummy Gates) i

- In $\mathscr{P}\mathfrak{lon}\mathscr{K}$, in the order for the protocol to be zero-knowledge, the authors add to the witness polynomials a blinding polynomial $b \in \mathbb{F}[X]$ as follows:

$$a(X) := b(X)Z_H(X) + \sum_{i=1}^{n} w_i \cdot L_i(X).$$

- This strategy ends up defining polynomials with degree $n + \deg(b)$, which is inefficient for practical scenarios in which n is a power of two.
- To avoid this issue, we instead sample $b_1, b_2 \in \mathbb{F}$ and compute:

$$a(X) := \sum_{i=1}^{n-2} w_i L_i(X) + b_1 L_{n-1}(X) + b_2 L_n(X).$$

Notice that now a has degree lower than n.

- However, for the permutation polynomial we do it in the standard way:

$$z(X) := (b_7X^2 + b_8X + b_9)Z_H(X) + L_1(X) + \sum_{i=1}^{n_1}\left(L_{i+1}(X)\prod_{j=1}^{i}\frac{(w_j + \beta\omega^j + \gamma)(w_{n+j} + \beta k_1\omega^j + \gamma)(w_{2n+j} + \beta k_2\omega^j + \gamma)}{(w_j + \beta\sigma^*(j) + \gamma)(w_{n+j} + \beta\sigma^*(n+j) + \gamma)(w_{2n+j} + \beta\sigma^*(2n+j) + \gamma)}\right)$$

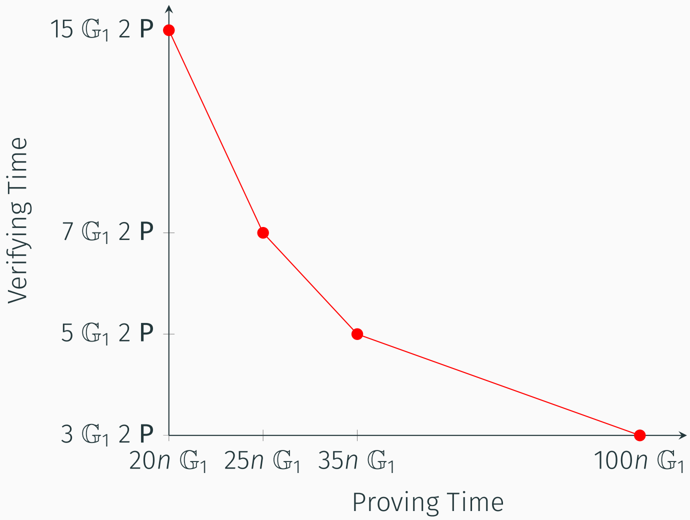
- In $\mathfrak{fflonK}$, every constraint adds an n factor to the prover time.
- If done with the dummy gates strategy, we would have needed to add the following constraint:

$$L_{n-1}(X)(z(X) - 1) = 0$$

to ensure the correctness of the permutation.

$\mathfrak{fflonK}^-$ and $\mathfrak{HyperfflonK}$

Tradeoff between $\mathcal{P}$ and $\mathcal{V}$ running times in $\mathfrak{fflonK}$



Thank you for your attention!

Dan Boneh, Justin Drake, Ben Fisch, and Ariel Gabizon.
Efficient polynomial commitment schemes for multiple points and polynomials.
Cryptology ePrint Archive, Report 2020/081, 2020.
https://eprint.iacr.org/2020/081.

Ariel Gabizon and Zachary J. Williamson.
fflonk: a fast-fourier inspired verifier efficient version of PlonK.
Cryptology ePrint Archive, Report 2021/1167, 2021.
https://eprint.iacr.org/2021/1167.

Yupeng Zhang, Daniel Genkin, Jonathan Katz, Dimitrios Papadopoulos, and Charalampos Papamanthou.
vRAM: Faster verifiable RAM with program-independent preprocessing.
In *2018 IEEE Symposium on Security and Privacy*, pages 908–925. IEEE Computer Society Press, May 2018.