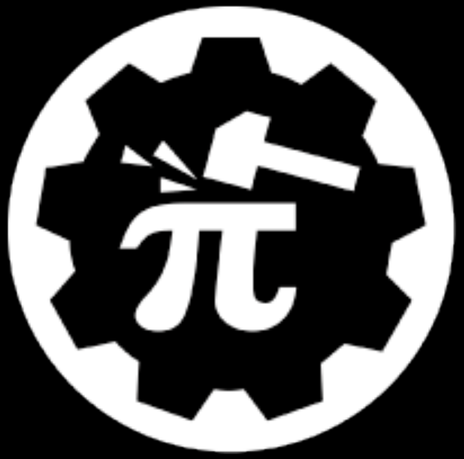
Gemini

Elastic proofs for diverse environments

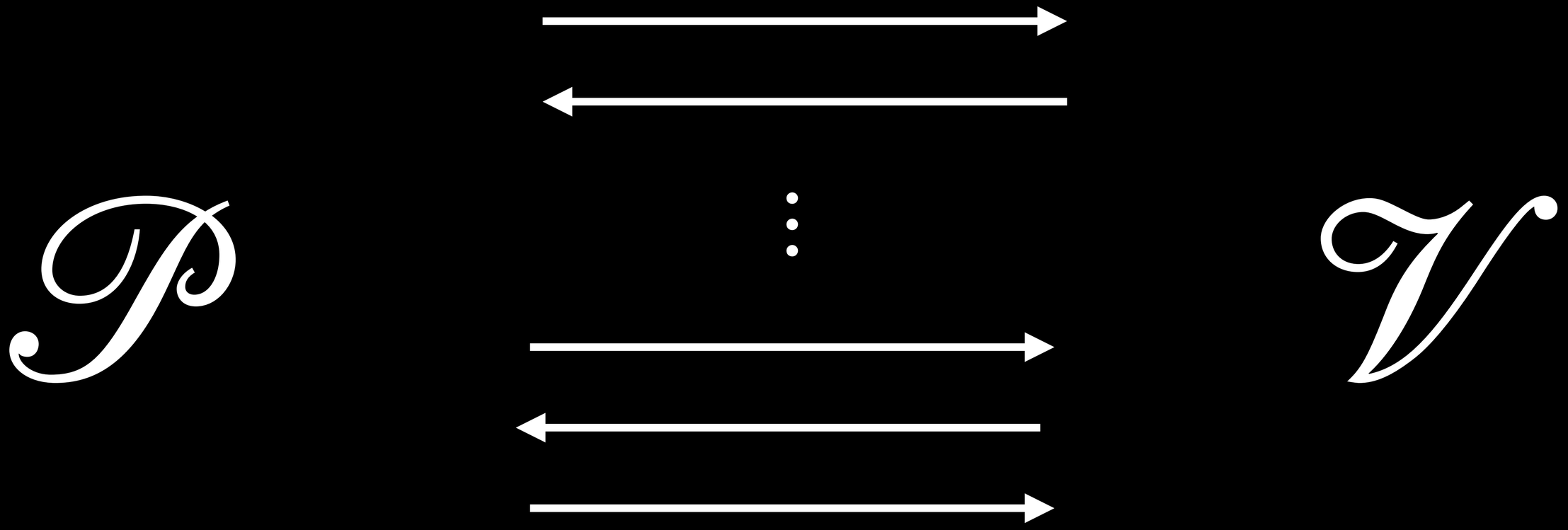
Joint work with Jonathan Bootle, Alessandro Chiesa, Yuncong Hu



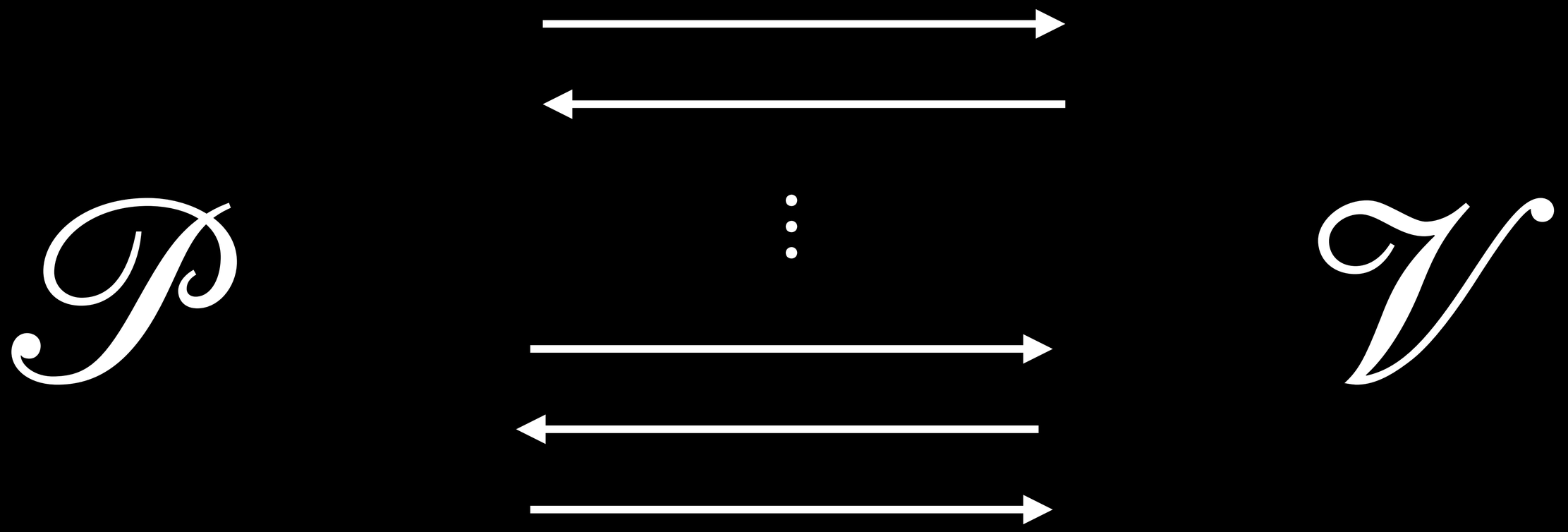
ia.cr/2022/420

arkworks.rs/gemini

Succinct arguments



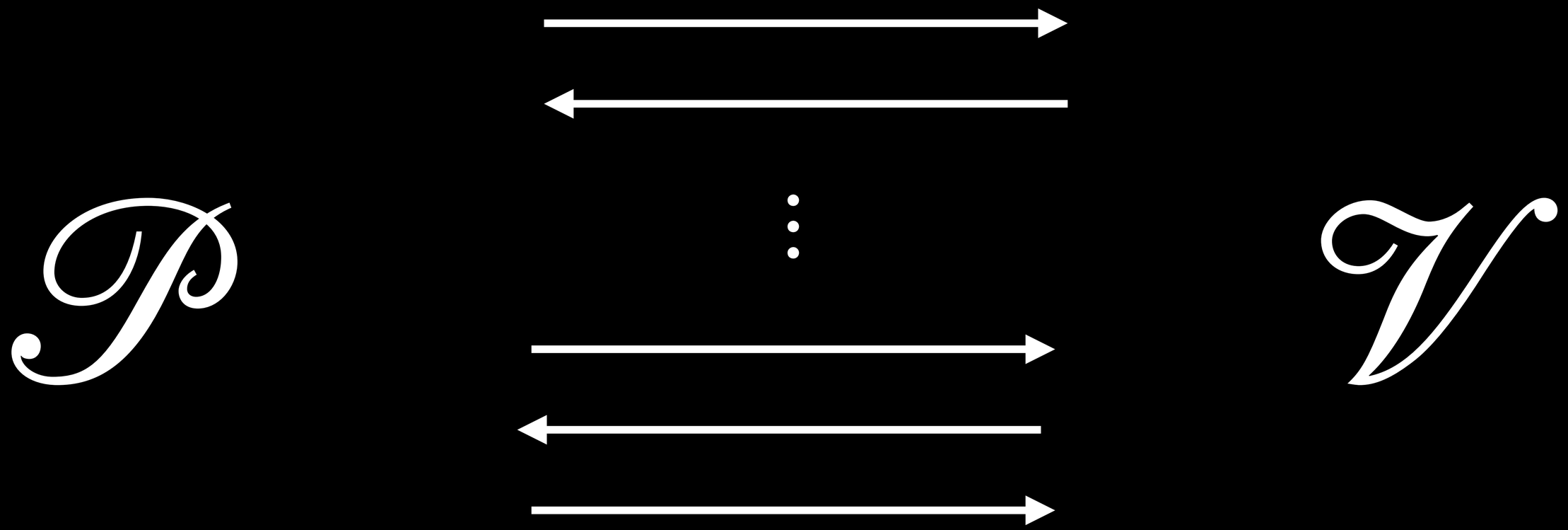
Succinct arguments



$\exists w \;.\; (x, w) \in \mathsf{R}$

Succinct arguments

(with preprocessing)



$\exists w \ . \ (x, w) \in \mathsf{R}$

Are we done?

Today, the biggest bottleneck is space.

Are we done?

Today, the biggest bottleneck is space.

- Proving time: $\tilde{O}(n)$

Are we done?

Today, the biggest bottleneck is space.

- Proving time: $\tilde{O}(n)$
- Verification time: $\tilde{O}(1)$

Are we done?

Today, the biggest bottleneck is space.

- Proving time: $\tilde{O}(n)$
- Verification time: $\tilde{O}(1)$
- Proving space: $O(n)$

Are we done?

Today, the biggest bottleneck is space.

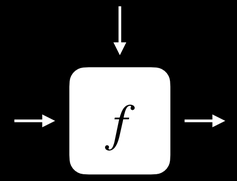
```
$ <prover> [billions of gate]
Killed
```

- Proving time: $\tilde{O}(n)$
- Verification time: $\tilde{O}(1)$
- Proving space: $O(n)$

An example

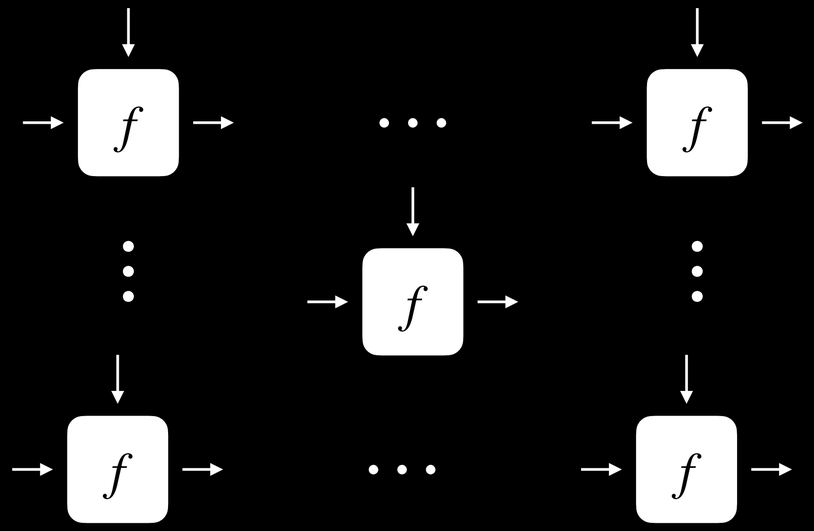
An example

- SHA256 compression function is ~30K constraints



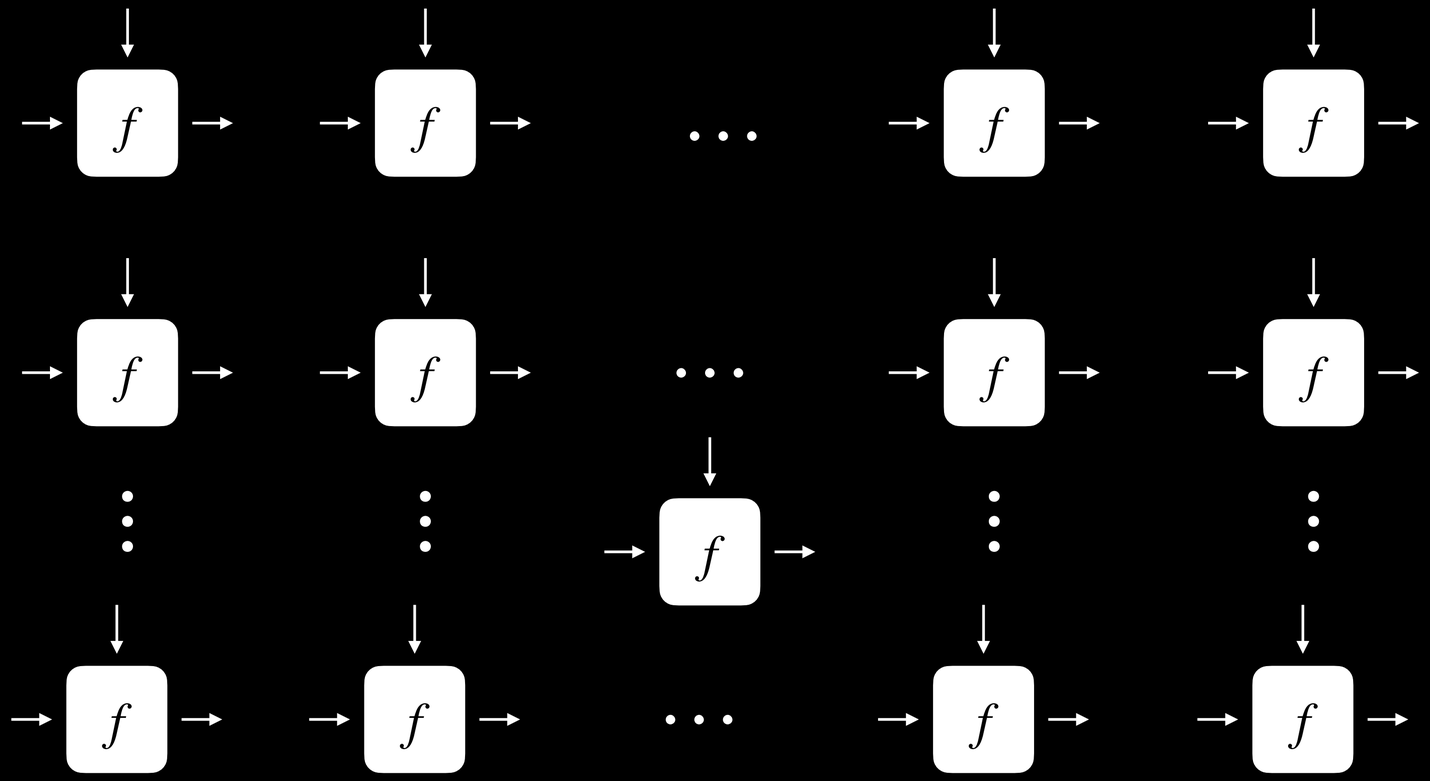
An example

- SHA256 compression function is ~30K constraints
- x100 is outside benchmarks in papers



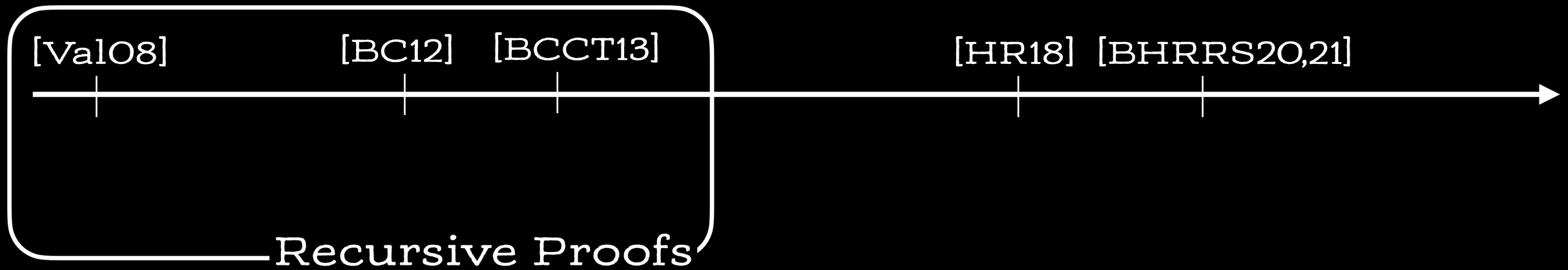
An example

- SHA256 compression function is ~30K constraints
- x100 is outside benchmarks in papers
- x1000 cannot be proven in modern implementations



Space-efficient arguments

Space-efficient arguments



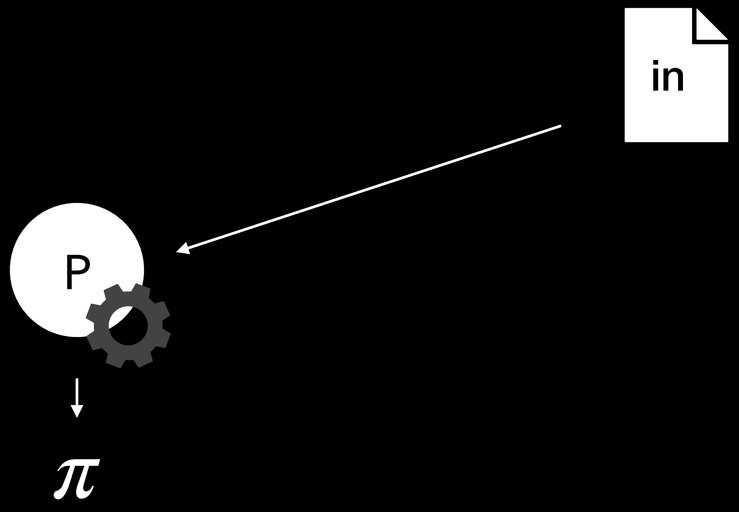
Streaming provers

Streaming provers

In practice

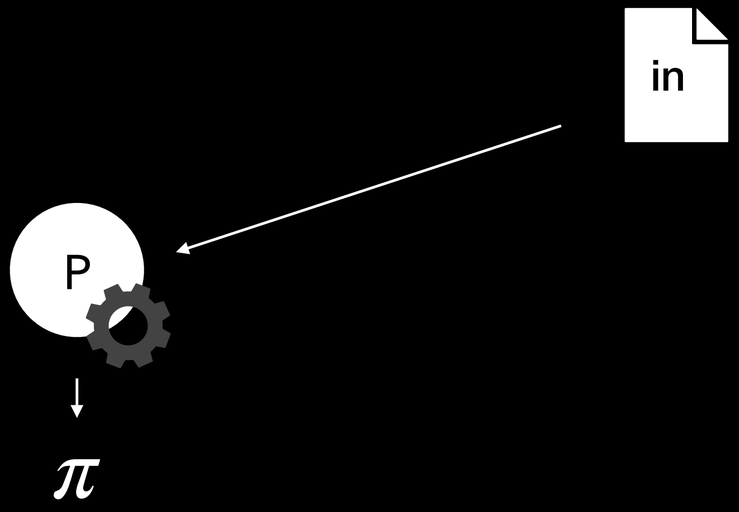
Streaming provers

In practice



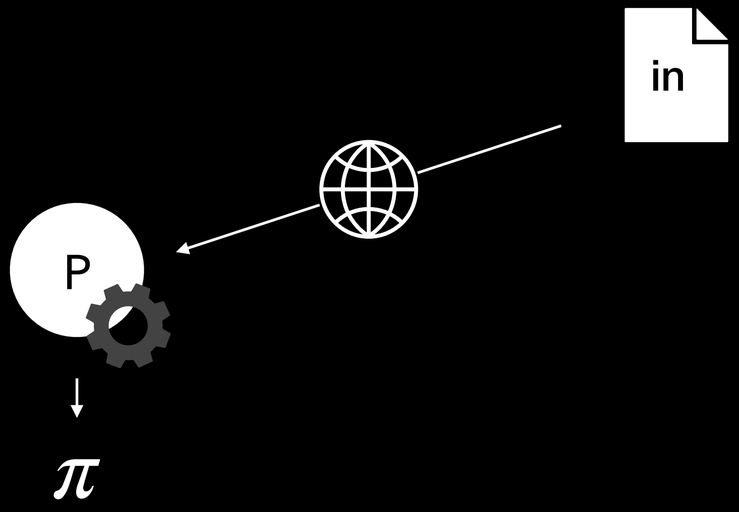
Streaming provers

In practice



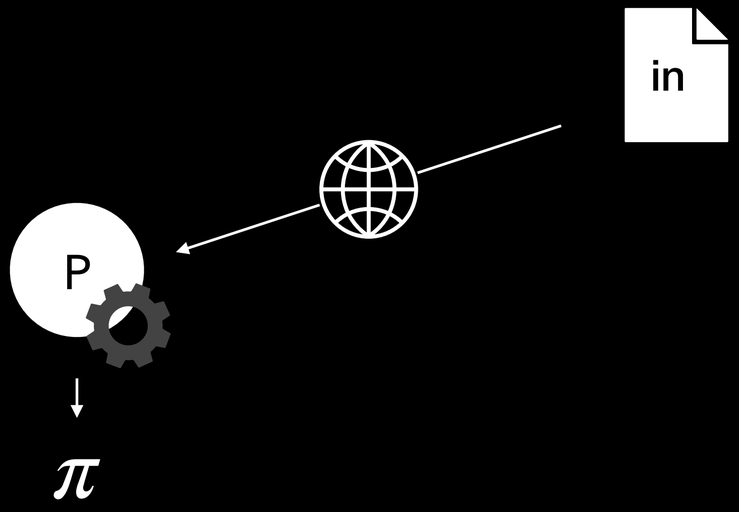
Streaming provers

In practice



Streaming provers

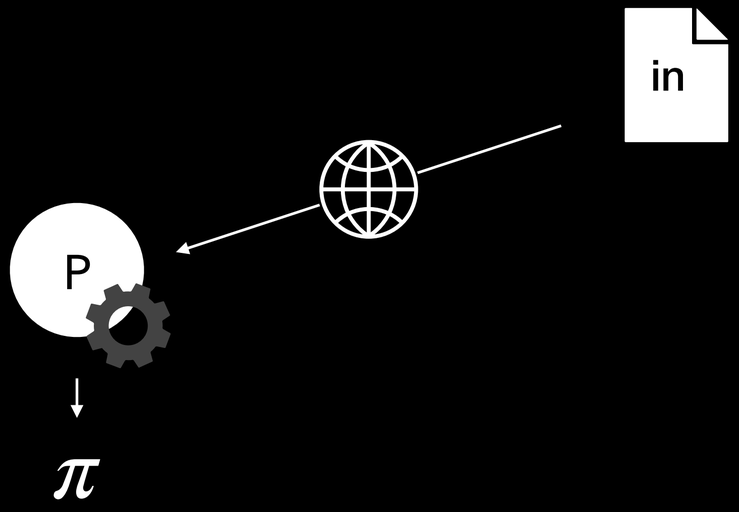
In practice



In theory

Streaming provers

In practice



In theory

$$\mathsf{P}^{\mathcal{S}(\text{in})} \rightarrow \pi$$

Elasticity

Our contribution

Two proving algorithms:

- Time-efficient 
- Space-efficient 

Same proof.

Main theorem

Our contribution

Thm. There exists an elastic (preprocessing) succinct argument for circuit satisfiability that admits two relalizations:

- A time-efficient prover that runs in $O(N)$ time and $O(N)$ space,
- A space-efficient prover that runs in $O(N \log^2 N)$ time and $O(\log N)$ space.

Main theorem

Our contribution

Thm. There exists an elastic (preprocessing) succinct argument for circuit satisfiability that admits two relalizations:

- A time-efficient prover that runs in $O(N)$ time and $O(N)$ space,
- A space-efficient prover that runs in $O(N \log^2 N)$ time and $O(\log N)$ space.

Main theorem

Our contribution

Thm. There exists an elastic (preprocessing) succinct argument for circuit satisfiability that admits two relalizations:

- A time-efficient prover that runs in $O(N)$ time and $O(N)$ space,
- A space-efficient prover that runs in $O(N \log^2 N)$ time and $O(\log N)$ space.

Proof is independent from the prover instantiation.

Verification runs in $O(\log N)$.

Main theorem

Our contribution

Thm. There exists an elastic (preprocessing) succinct argument for circuit satisfiability that admits two relalizations:

- A time-efficient prover that runs in $O(N)$ time and $O(N)$ space,
- A space-efficient prover that runs in $O(N \log^2 N)$ time and $O(\log N)$ space.

Proof is independent from the prover instantiation.

Verification runs in $O(\log N)$.

Rank-1 Constraint Systems

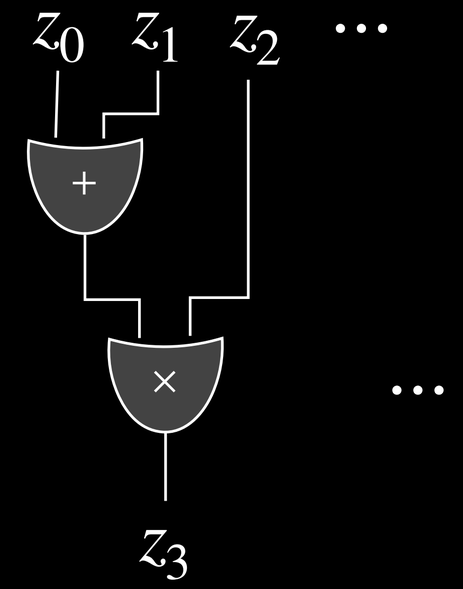
C circuit is satisfied

$$\Downarrow$$

$$\exists z \quad \text{s.t.} \quad Az \circ Bz = Cz$$

Rank-1 Constraint Systems

(n wires, m mult gates)



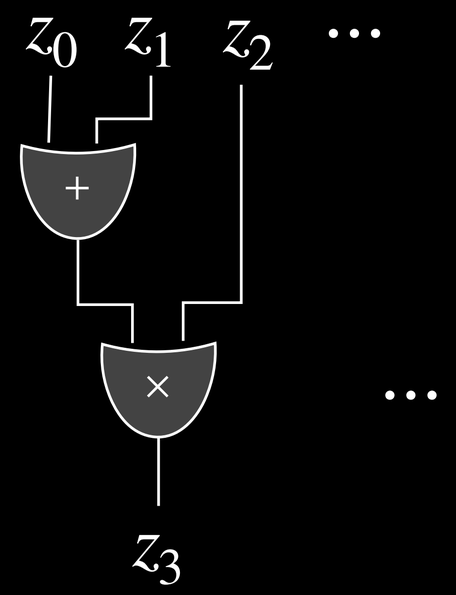
C circuit is satisfied

$$\Downarrow$$

$$\exists z \quad \text{s.t.} \quad Az \circ Bz = Cz$$

Rank-1 Constraint Systems

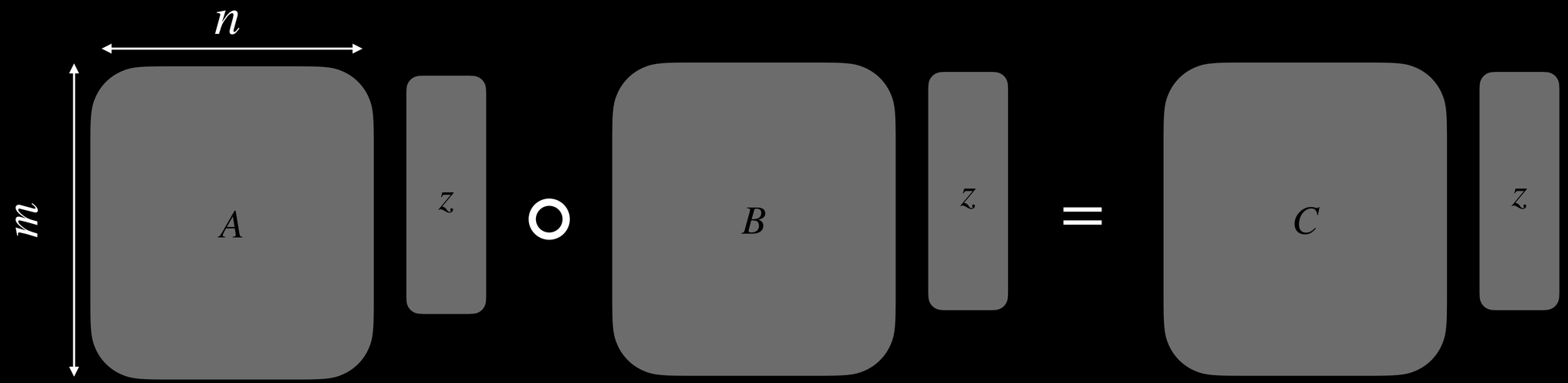
(n wires, m mult gates)



C circuit is satisfied

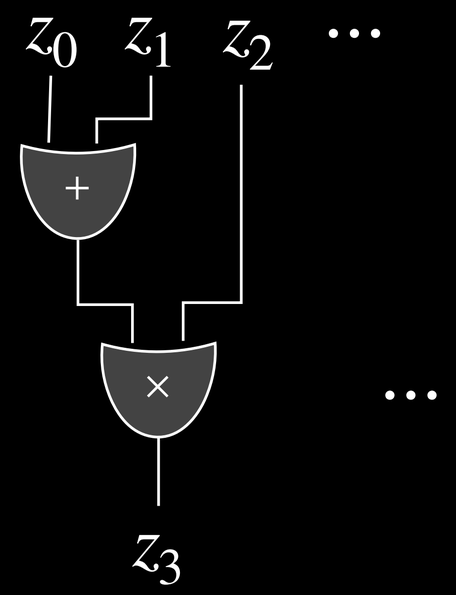
$$\Downarrow$$

$$\exists z \quad \text{s.t.} \quad Az \circ Bz = Cz$$



Rank-1 Constraint Systems

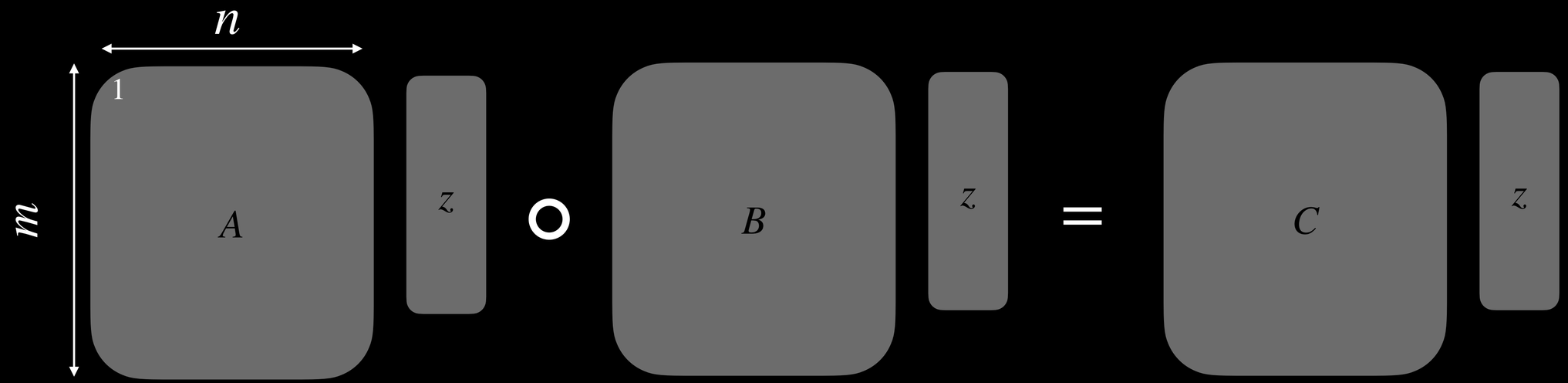
(n wires, m mult gates)



C circuit is satisfied

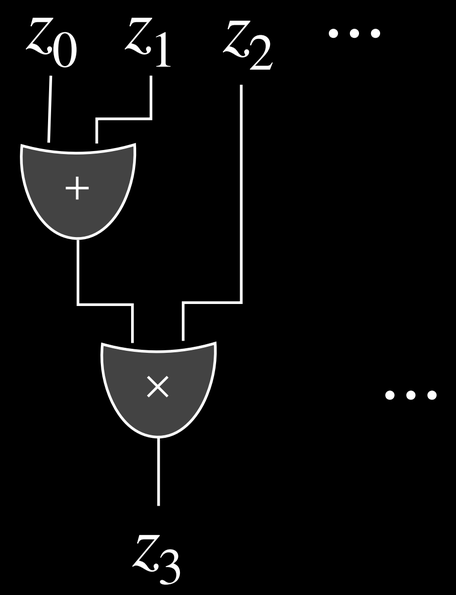
$$\Downarrow$$

$$\exists z \quad \text{s.t.} \quad Az \circ Bz = Cz$$



Rank-1 Constraint Systems

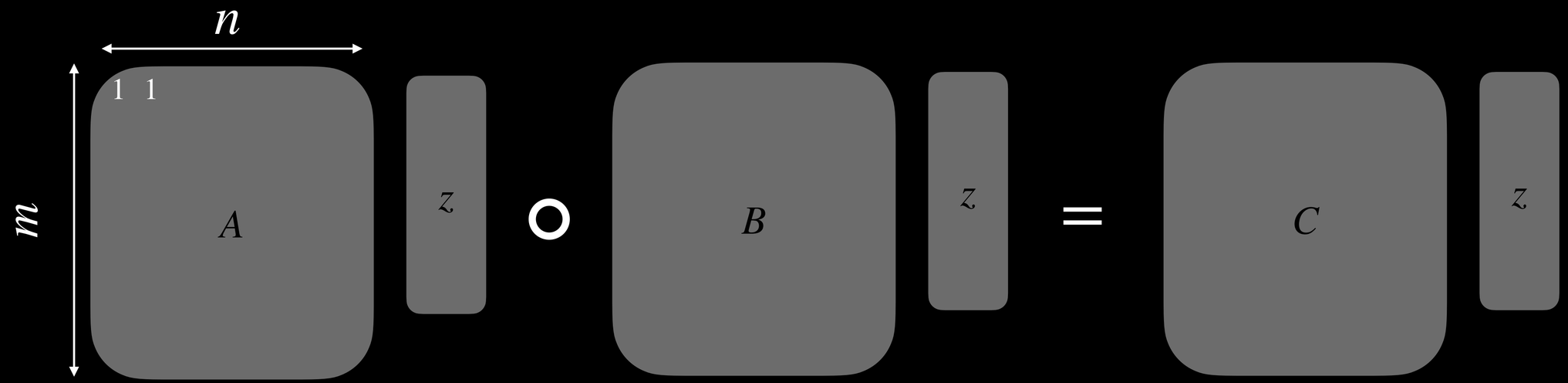
(n wires, m mult gates)



C circuit is satisfied

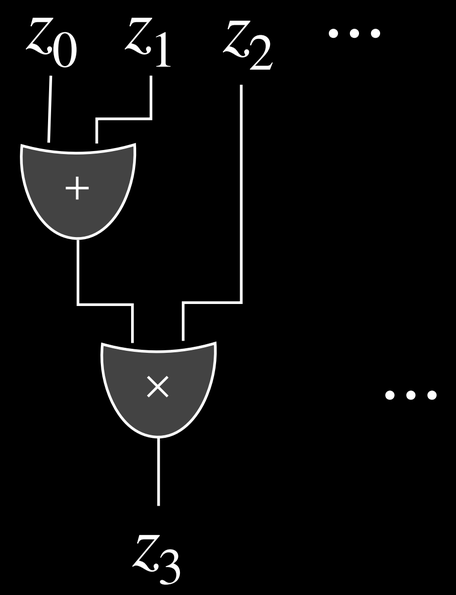
$$\Downarrow$$

$$\exists z \quad \text{s.t.} \quad Az \circ Bz = Cz$$



Rank-1 Constraint Systems

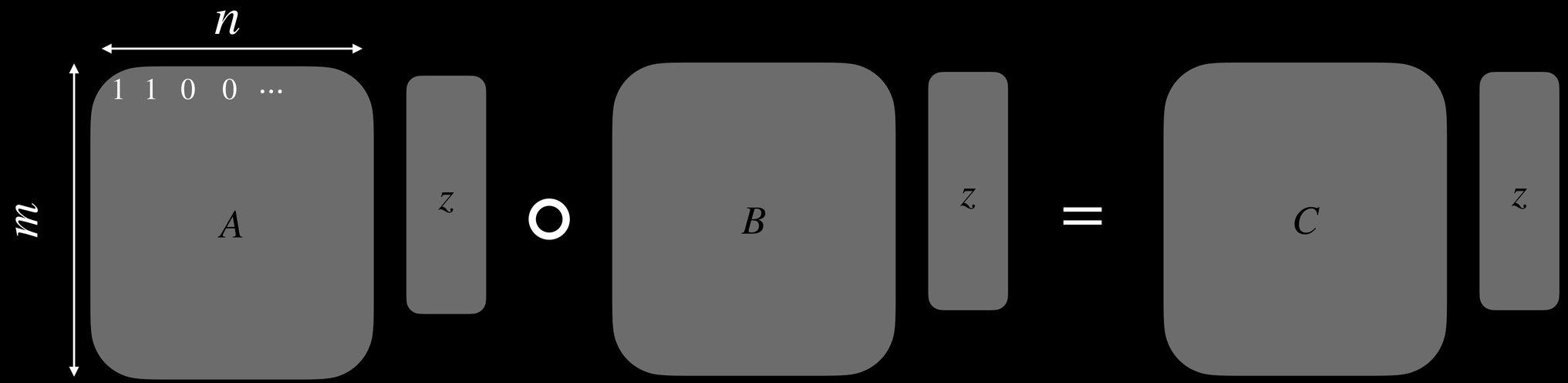
(n wires, m mult gates)



C circuit is satisfied

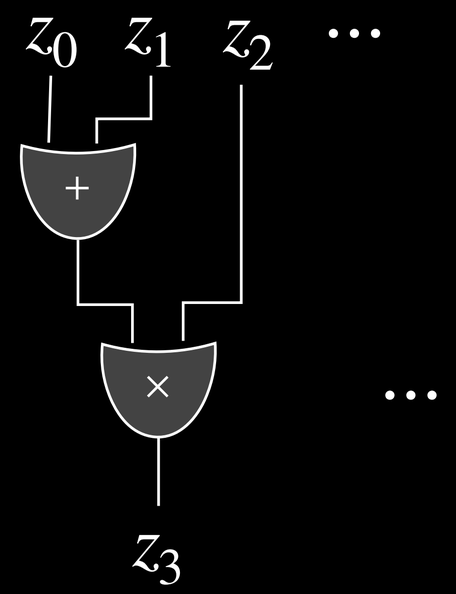
$$\Downarrow$$

$$\exists z \quad \text{s.t.} \quad Az \circ Bz = Cz$$



Rank-1 Constraint Systems

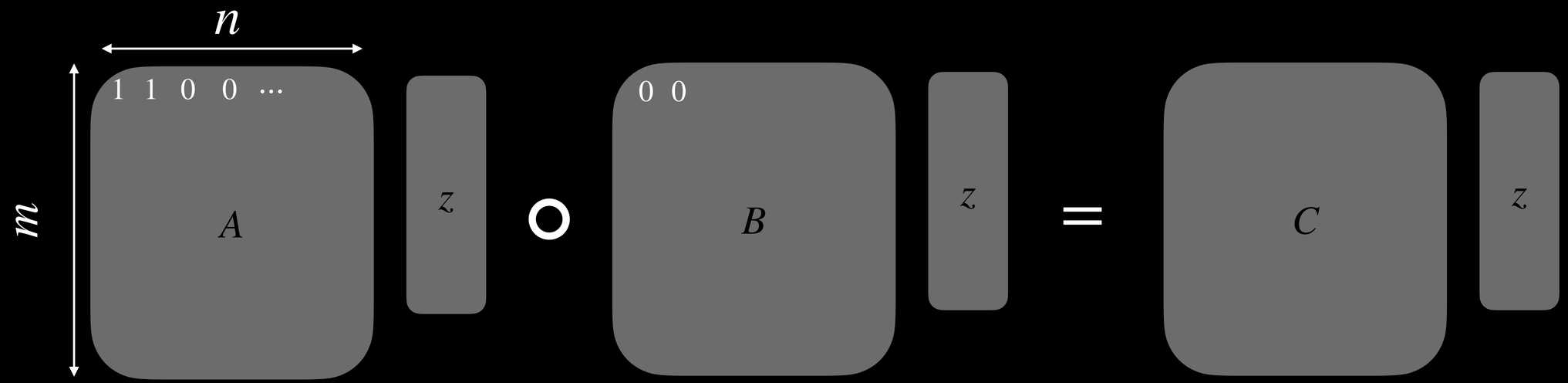
(n wires, m mult gates)



C circuit is satisfied

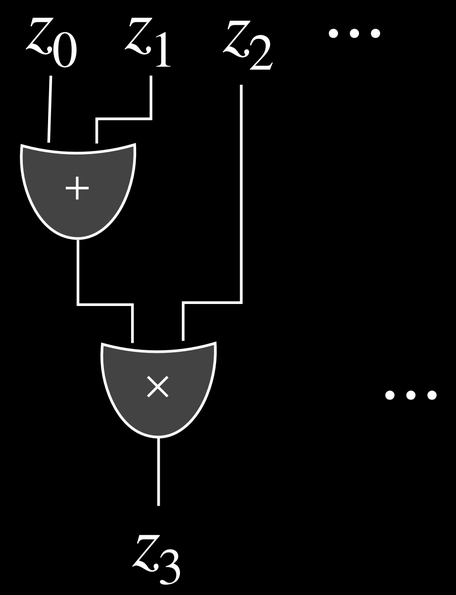
$$\Downarrow$$

$$\exists z \quad \text{s.t.} \quad Az \circ Bz = Cz$$



Rank-1 Constraint Systems

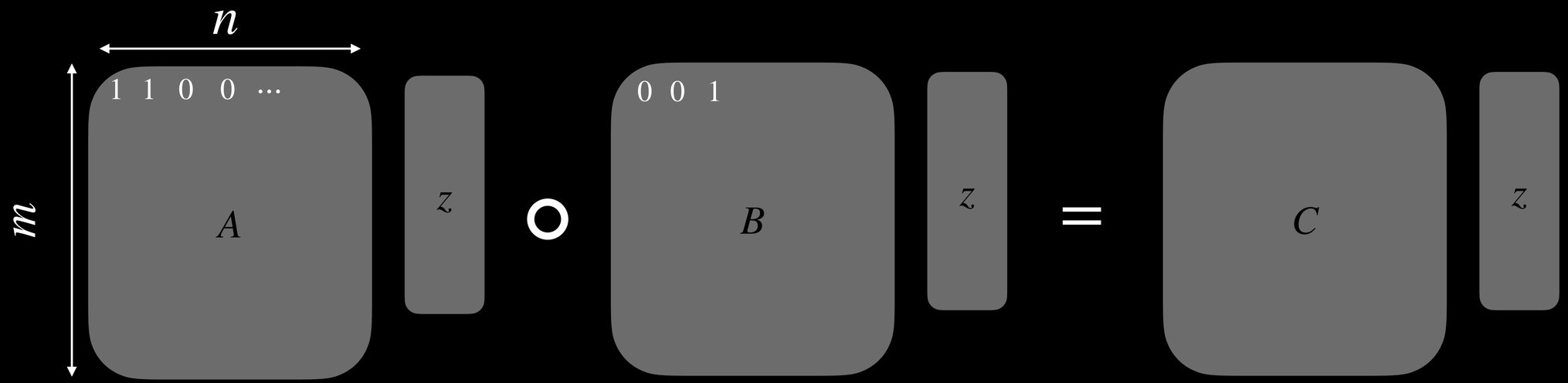
(n wires, m mult gates)



$\mathbb{C}$ circuit is satisfied

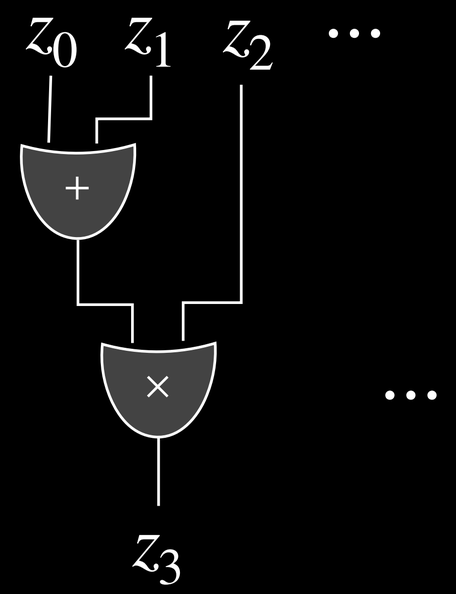
$$\Downarrow$$

$$\exists z \quad \text{s.t.} \quad Az \circ Bz = Cz$$



Rank-1 Constraint Systems

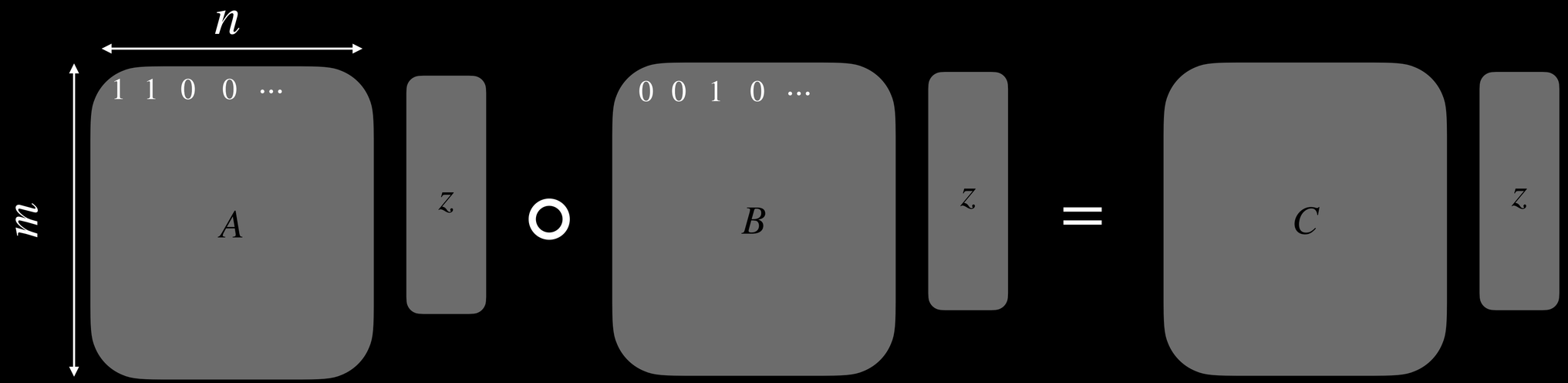
(n wires, m mult gates)



C circuit is satisfied

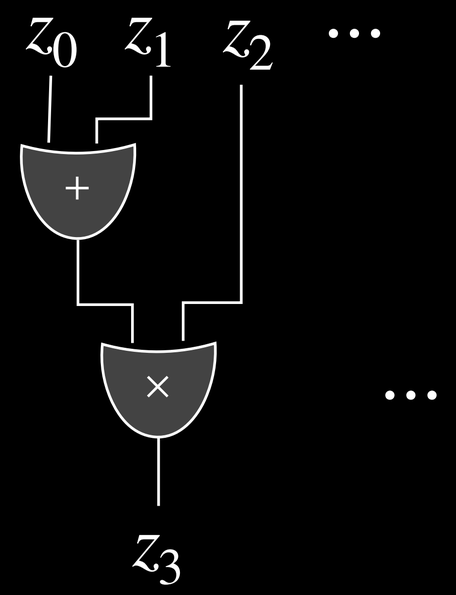
$$\Downarrow$$

$$\exists z \quad \text{s.t.} \quad Az \circ Bz = Cz$$



Rank-1 Constraint Systems

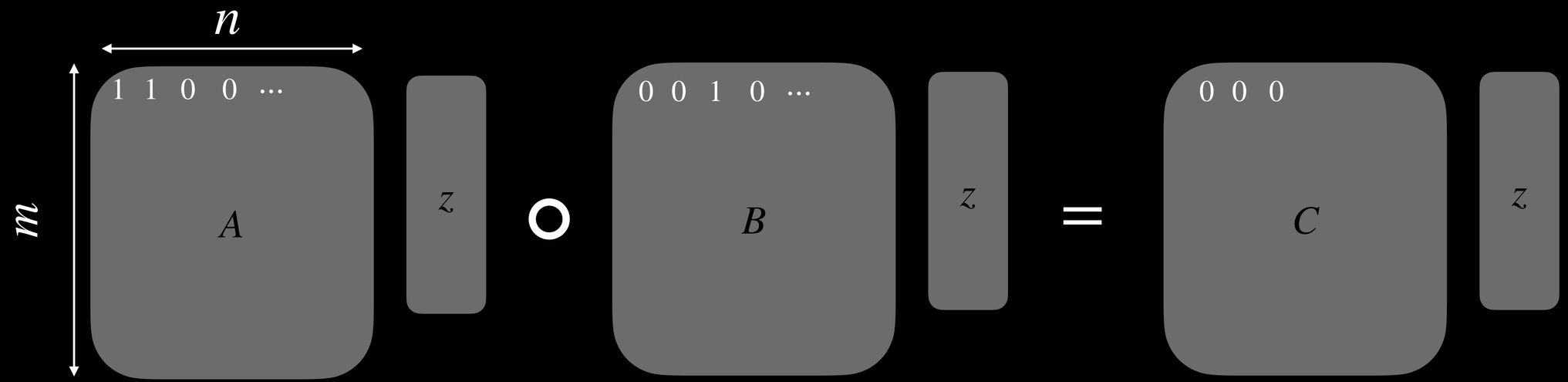
(n wires, m mult gates)



C circuit is satisfied

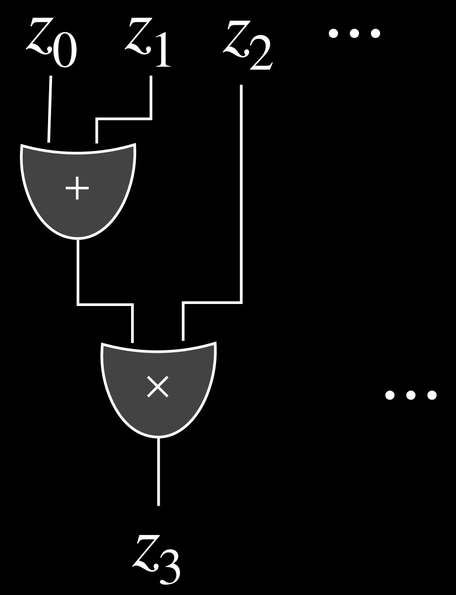
$$\Downarrow$$

$$\exists z \quad \text{s.t.} \quad Az \circ Bz = Cz$$



Rank-1 Constraint Systems

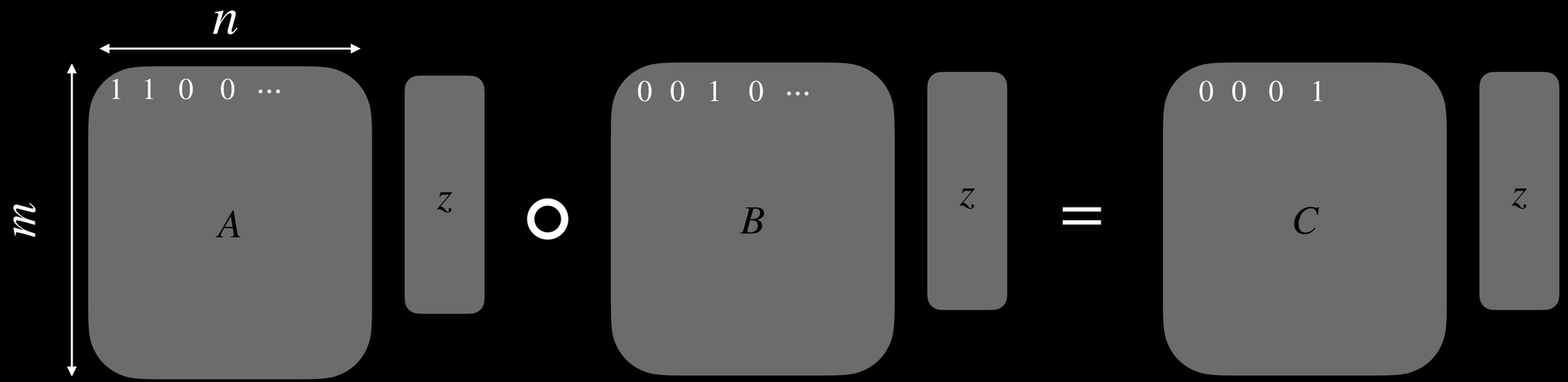
(n wires, m mult gates)



C circuit is satisfied

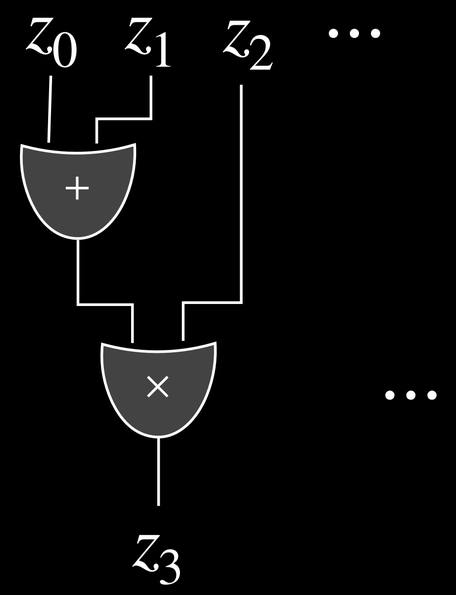
$$\Downarrow$$

$$\exists z \quad \text{s.t.} \quad Az \circ Bz = Cz$$



Rank-1 Constraint Systems

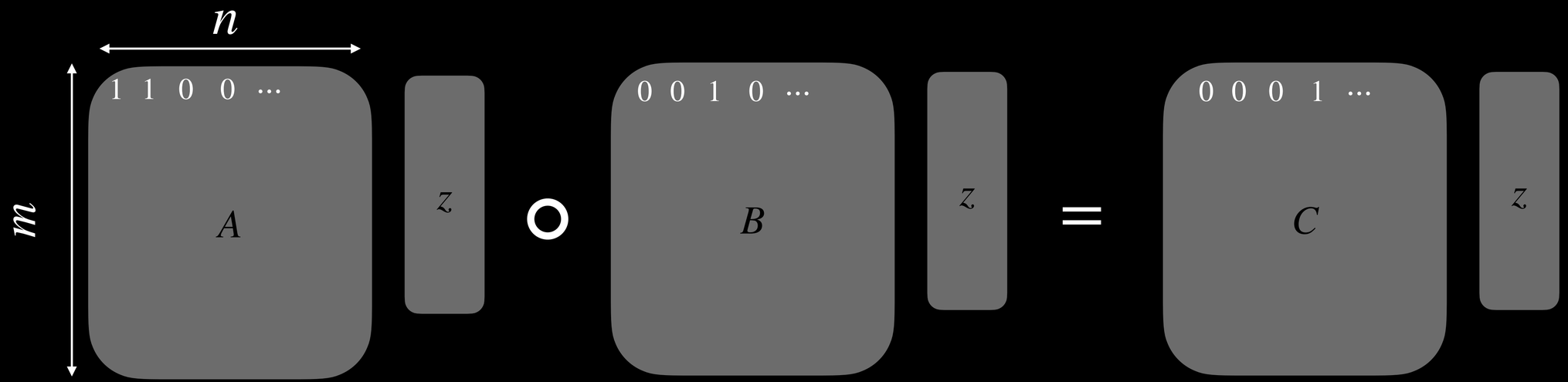
(n wires, m mult gates)



C circuit is satisfied

$$\Downarrow$$

$$\exists z \quad \text{s.t.} \quad Az \circ Bz = Cz$$



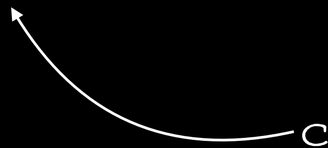
Streaming R1CS

Our contribution

$$\exists z \quad \text{s.t.} \quad Az \circ Bz = Cz$$

Def. The stream of a R1CS relation consists of:

- the full instance $\mathcal{S}(z)$,
- the matrices $\mathcal{S}_{\text{colmaj}}(A)$, $\mathcal{S}_{\text{rowmaj}}(A)$, etc.,
- the computation trace $\mathcal{S}(Az)$, $\mathcal{S}(Bz)$, $\mathcal{S}(Cz)$.



Can be generated composing streams!

Idealized Protocol

Idealized Protocol

An Elastic Polynomial Interactive Oracle Proof for Inner Product

Inner product argument

Inner product argument

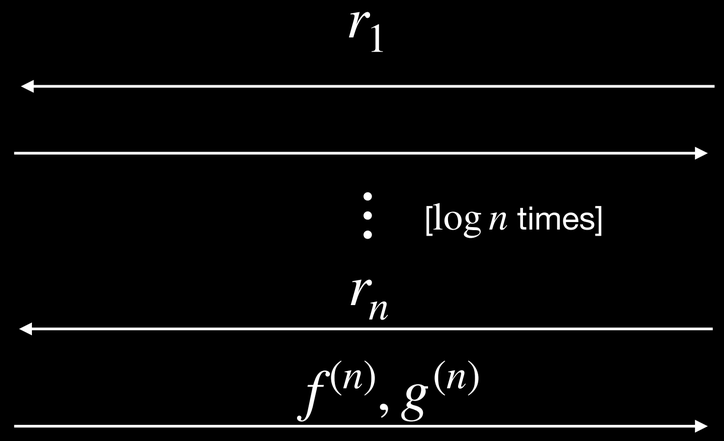
$$\langle \vec{f}, \overrightarrow{g} \rangle = y \text{ of size } n \text{ is valid} \overset{\text{whp}}{\Longrightarrow} \langle \vec{f}', \overrightarrow{g}' \rangle = y' \text{ of size } n/2 \text{ is valid}$$

Inner product argument

$\langle \vec{f}, \vec{g} \rangle = y$ of size n is valid $\overset{\text{whp}}{\Longrightarrow}$ $\langle \vec{f}', \vec{g}' \rangle = y'$ of size $n/2$ is valid

Folding:

- $\vec{f}' = \vec{f}_{\text{even}} + r_1 \cdot \vec{f}_{\text{odd}}$



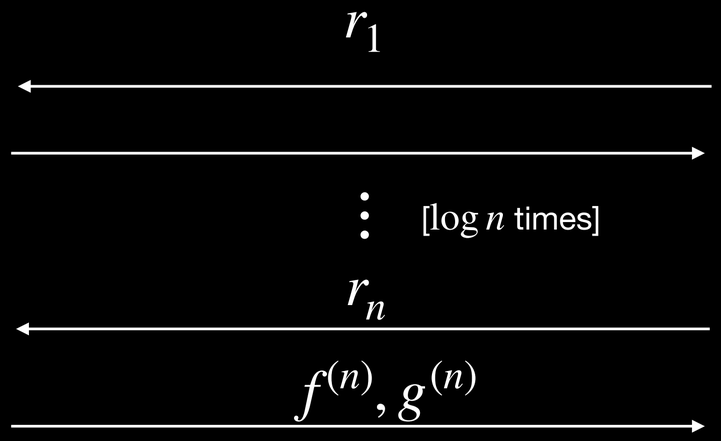
$r_j \leftarrow \mathbb{F}_p$

Inner product argument

$$\langle \vec{f}, \vec{g} \rangle = y \text{ of size } n \text{ is valid} \overset{\text{whp}}{\Longrightarrow} \langle \vec{f}', \vec{g}' \rangle = y' \text{ of size } n/2 \text{ is valid}$$

Folding:

- $\vec{f}' = \vec{f}_{\text{even}} + r_1 \cdot \vec{f}_{\text{odd}}$
- $\vec{f}' = \vec{f}_{\text{left}} + r_1 \cdot \vec{f}_{\text{right}}$



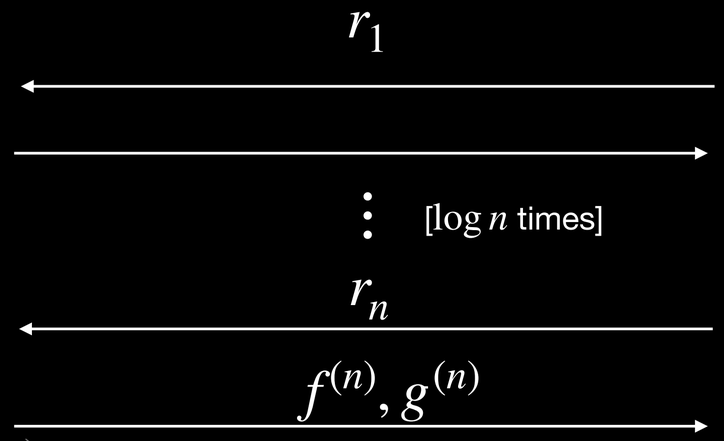
$r_j \leftarrow \mathbb{F}_p$

Inner product argument

$$\langle \vec{f}, \vec{g} \rangle = y \text{ of size } n \text{ is valid} \quad \overset{\text{whp}}{\Longrightarrow} \quad \langle \vec{f}', \vec{g}' \rangle = y' \text{ of size } n/2 \text{ is valid}$$

Folding:

- $\vec{f}' = \vec{f}_{\text{even}} + r_1 \cdot \vec{f}_{\text{odd}}$
- $\vec{f}' = \vec{f}_{\text{left}} + r_1 \cdot \vec{f}_{\text{right}}$



$r_j \leftarrow \mathbb{F}_p$

$$\vec{f}'' = \vec{f}_{\text{e,e}} + r_1 \cdot \vec{f}_{\text{e,o}} + r_2 \cdot \vec{f}_{\text{o,e}} + r_1 r_2 \cdot \vec{f}_{\text{o,o}}$$

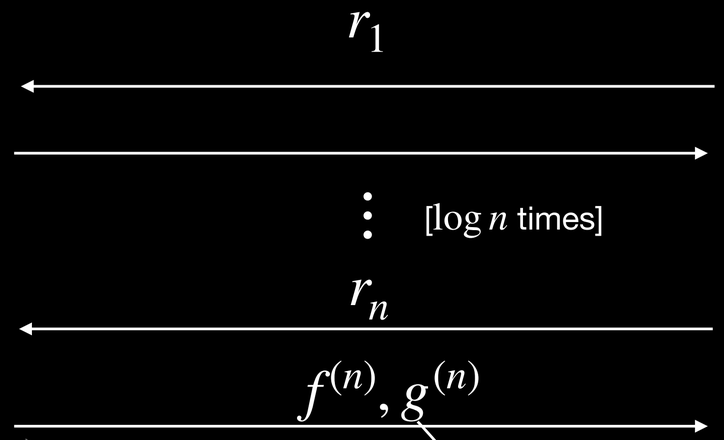
Inner product argument

$\langle \vec{f}, \vec{g} \rangle = y$ of size n is valid $\overset{\text{whp}}{\Longrightarrow}$ $\langle \vec{f}', \vec{g}' \rangle = y'$ of size $n/2$ is valid

Folding:

- $\vec{f}' = \vec{f}_{\text{even}} + r_1 \cdot \vec{f}_{\text{odd}}$
- $\vec{f}' = \vec{f}_{\text{left}} + r_1 \cdot \vec{f}_{\text{right}}$

$\vec{f}'' = \vec{f}_{\text{e,e}} + r_1 \cdot \vec{f}_{\text{e,o}} + r_2 \cdot \vec{f}_{\text{o,e}} + r_1 r_2 \cdot \vec{f}_{\text{o,o}}$



$r_j \leftarrow \mathbb{F}_p$

$f^{(n)}, g^{(n)}$ are a multivariate polynomial evaluation

Streaming the prover's messages

Our contribution

Stream of f:

Streaming the prover's messages

Our contribution

Stream of f:

Streaming the prover's messages

Our contribution

Stream of f:

Streaming the prover's messages

Our contribution

Stream of f:

Streaming the prover's messages

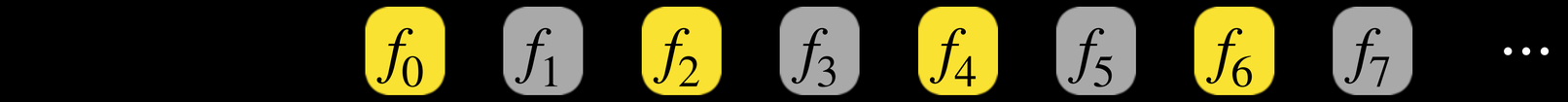
Our contribution

Stream of f:

Streaming the prover's messages

Our contribution

Stream of f:



Streaming the prover's messages

Our contribution

Stream of f:

Stack

Streaming the prover's messages

Our contribution

Stream of f:

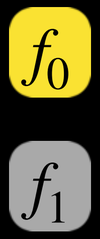
Stack

Streaming the prover's messages

Our contribution

Stream of f:

Stack



Streaming the prover's messages

Our contribution

Stream of f:

f_2 f_3 f_4 f_5 f_6 f_7 ...

Stack

$f'_0 := f_0 + r_1 \cdot f_1$

Streaming the prover's messages

Our contribution

Stream of f:

Stack

Streaming the prover's messages

Our contribution

Stream of f:

f_3 f_4 f_5 f_6 f_7 ...

Stack

f'_0

f_2

Streaming the prover's messages

Our contribution

Stream of f:

Stack

Streaming the prover's messages

Our contribution

Stream of f:

f_4 f_5 f_6 f_7 ...

Stack

f'_0

$f'_1 := f_2 + r_1 \cdot f_3$

Streaming the prover's messages

Our contribution

Stream of f:

Stack

Streaming the prover's messages

Our contribution

Stream of f:

Stack

$f''_0 := f'_0 + r_2 \cdot f'_1$

Streaming the prover's messages

Our contribution

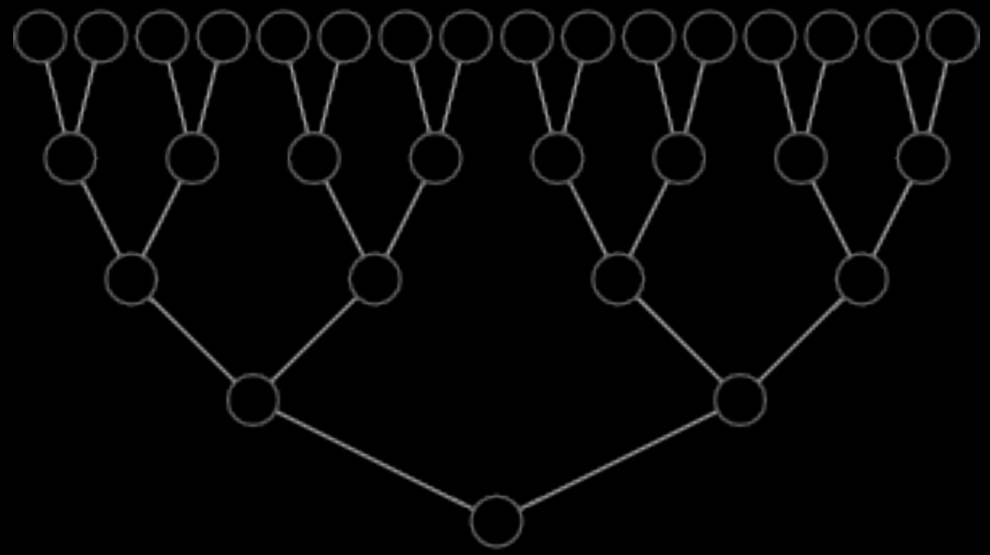
Stream of f:

f_4 f_5 f_6 f_7 ...

Stack

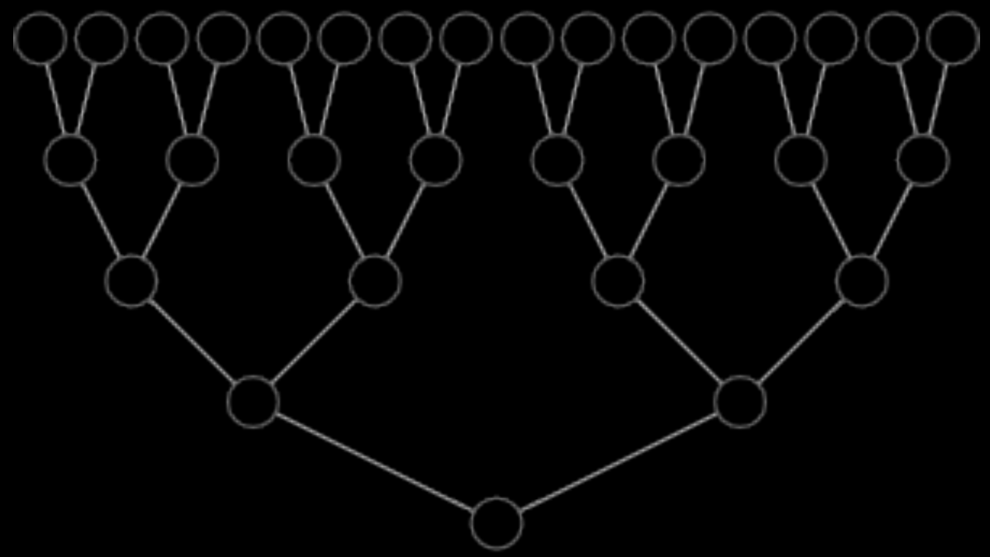
$f_0'' := f_0' + r_2 \cdot f_1'$; yield f_0'' [the first coefficient!]

Inner product: complexity



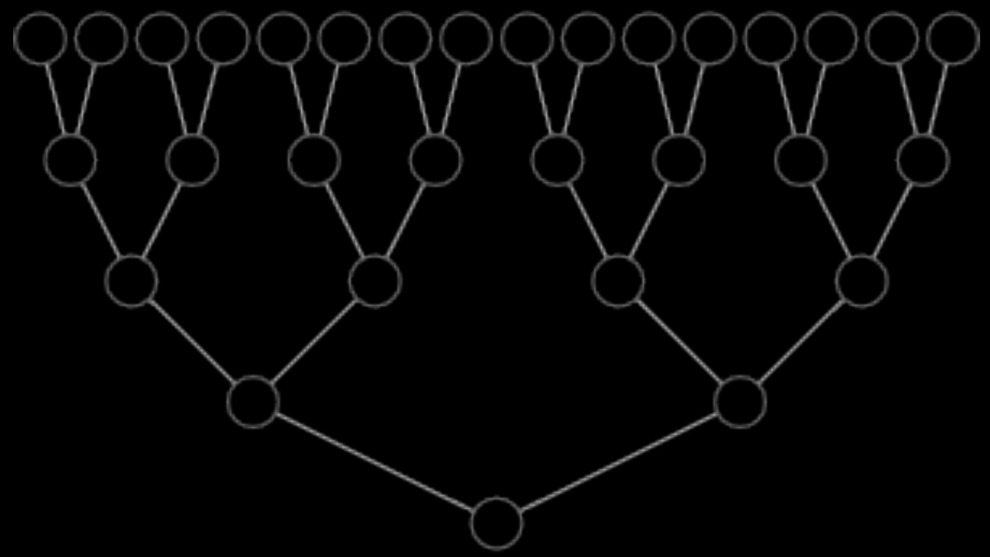
Inner product: complexity

- The proving algorithm can use $\log n$ space
- .. and overall $n \log n$ time.



Inner product: complexity

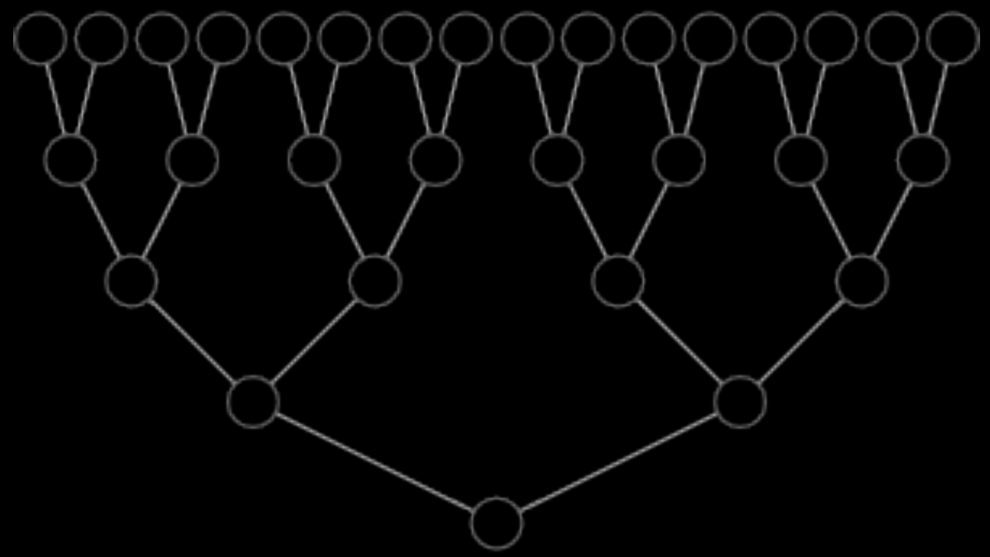
- The proving algorithm can use $\log n$ space
- .. and overall $n \log n$ time.



Exploit divide and conquer structure:

Inner product: complexity

- The proving algorithm can use $\log n$ space
- .. and overall $n \log n$ time.

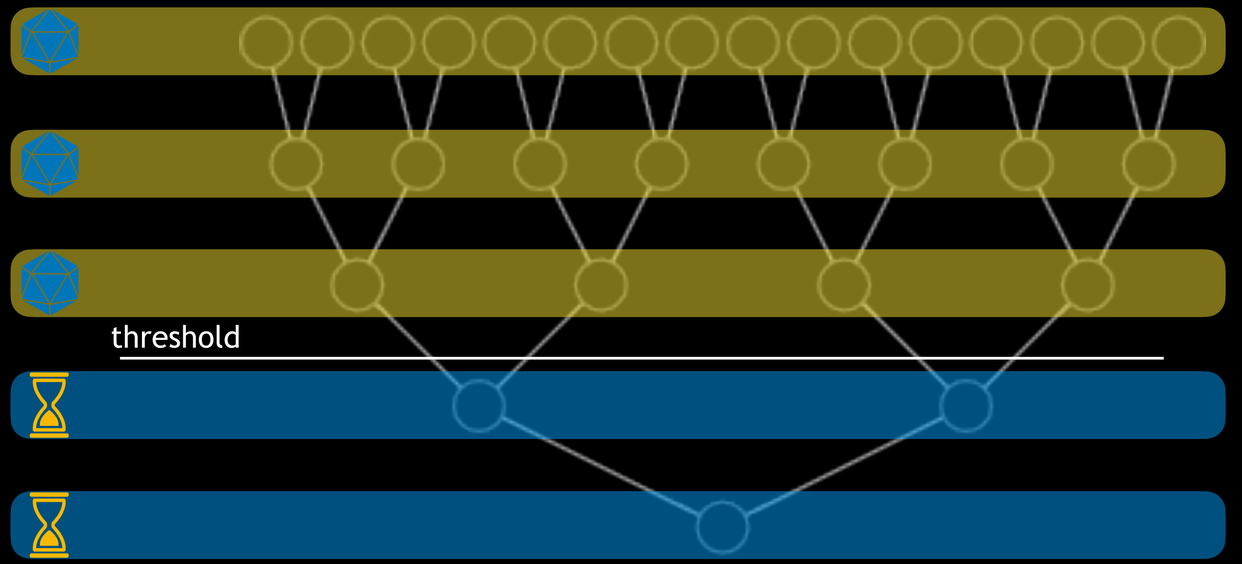


Exploit divide and conquer structure:

- Fix a memory budget

Inner product: complexity

- The proving algorithm can use $\log n$ space
- .. and overall $n \log n$ time.

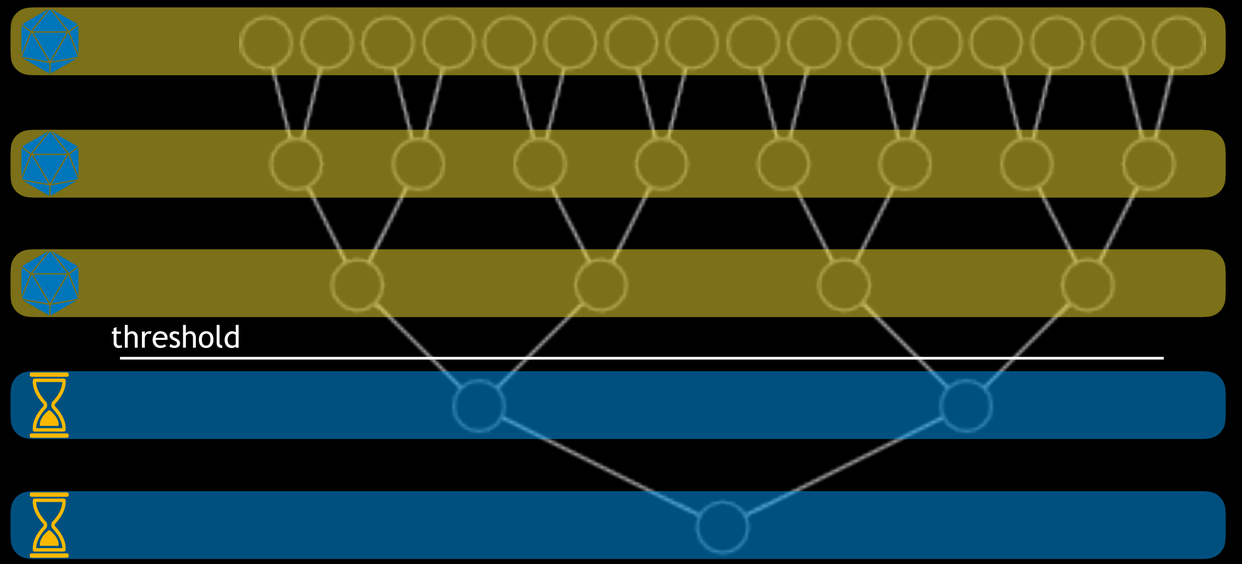


Exploit divide and conquer structure:

- Fix a memory budget

Inner product: complexity

- The proving algorithm can use $\log n$ space
- .. and overall $n \log n$ time.

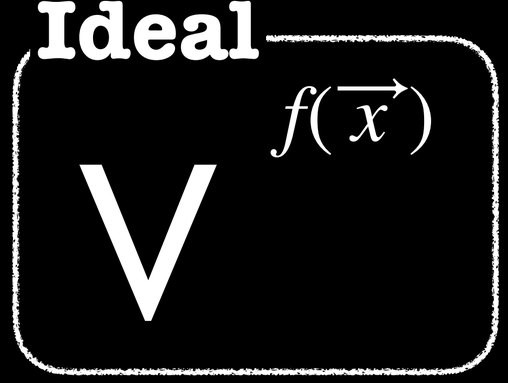


Exploit divide and conquer structure:

- Fix a memory budget
- Overall complexity $O(n \log n)$ for arbitrary size, $O(n)$ from threshold

Tensorcheck

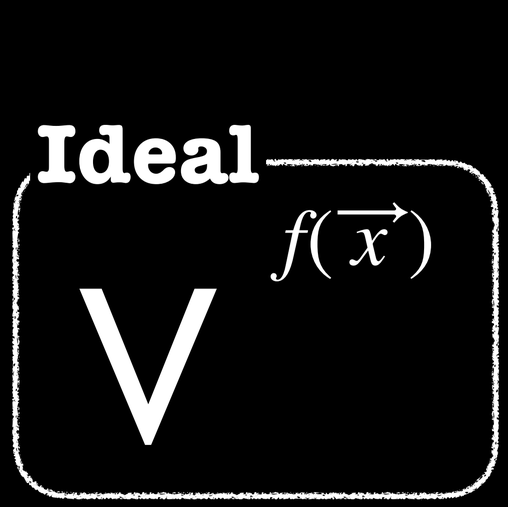
At the end of inner product: $f(r_1, r_2, \ldots, r_n) = f^{(n)}$

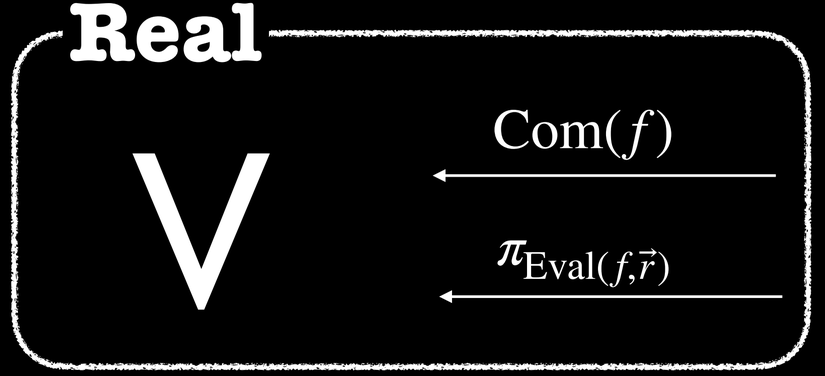


Probabilistic Interactive Oracle Proofs
with Multivariate Polynomial queries

Tensorcheck

At the end of inner product: $f(r_1, r_2, \ldots, r_n) = f^{(n)}$



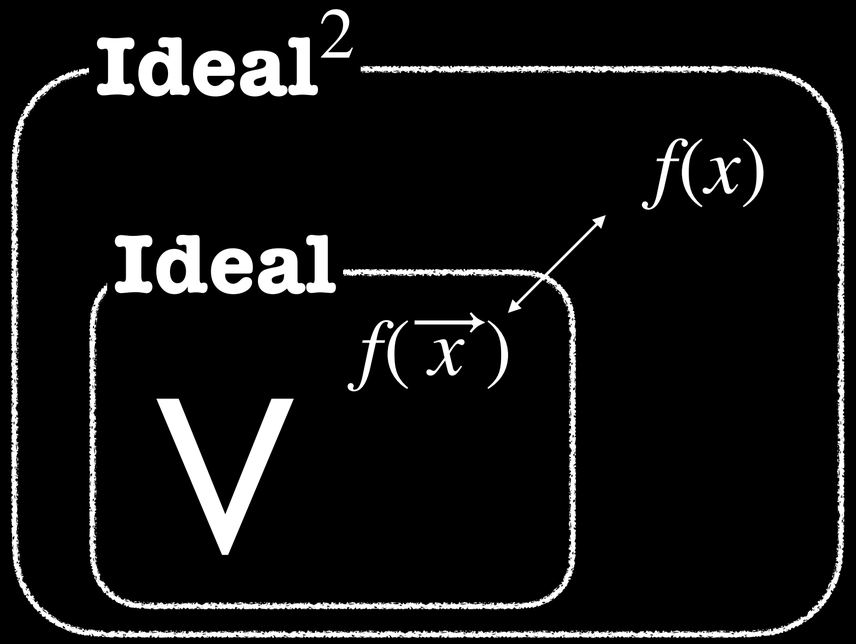


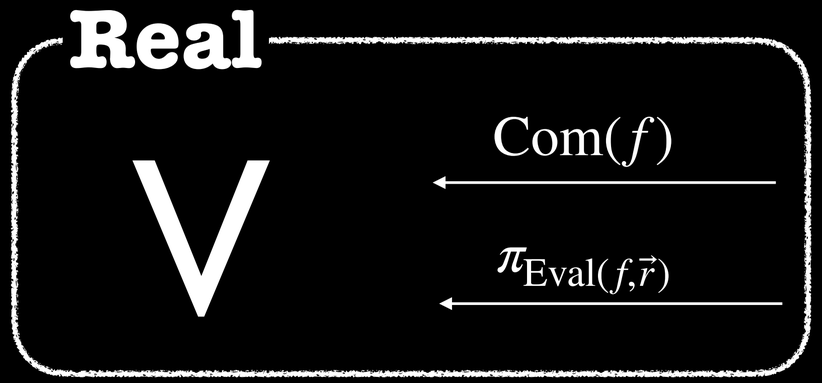
Probabilistic Interactive Oracle Proofs
with Multivariate Polynomial queries

Multivariate
Polynomial Commitment Scheme

Tensorcheck

At the end of inner product: $f(r_1, r_2, \ldots, r_n) = f^{(n)}$



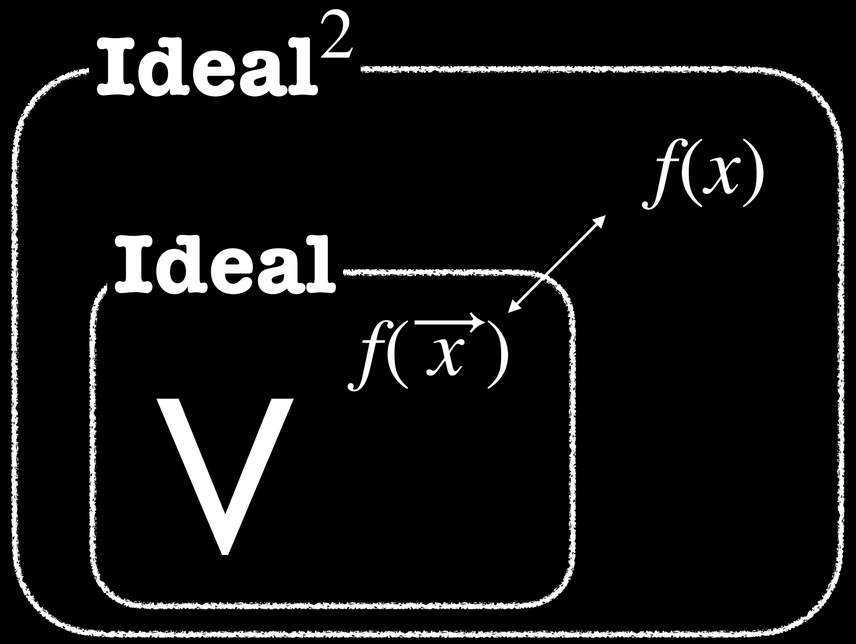


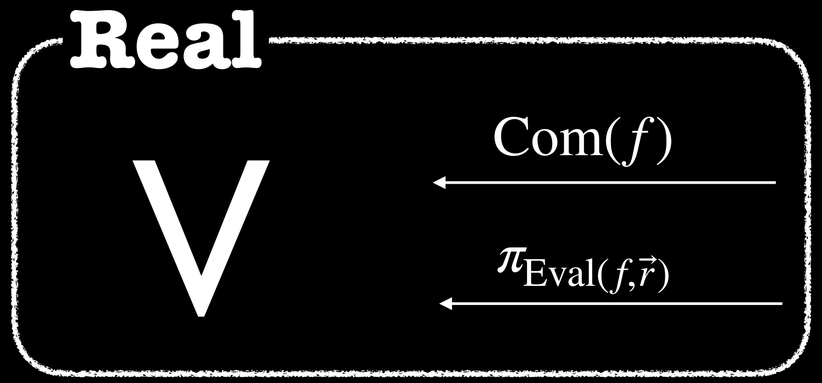
Probabilistic Interactive Oracle Proofs
with Multivariate Polynomial queries

Multivariate
Polynomial Commitment Scheme

Tensorcheck

At the end of inner product: $f(r_1, r_2, \ldots, r_n) = f^{(n)}$



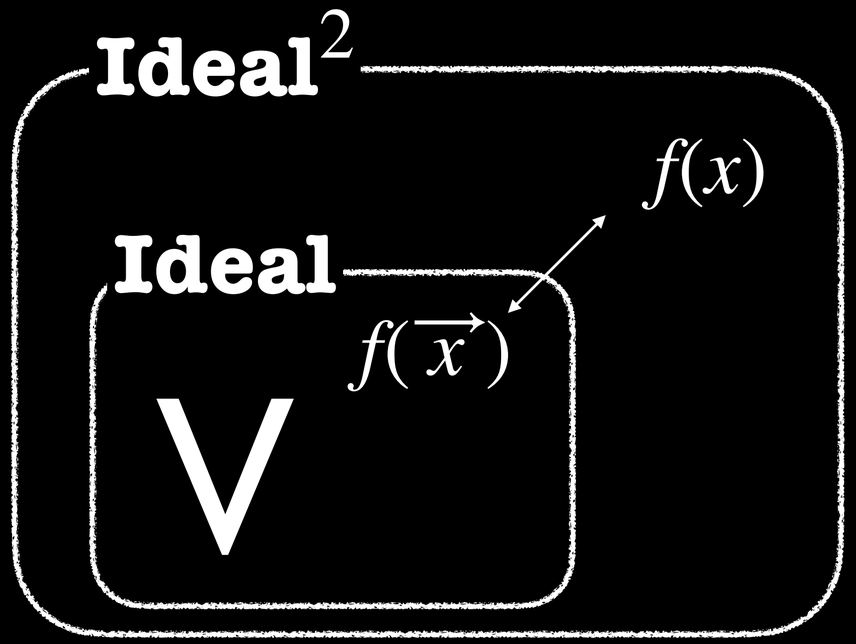


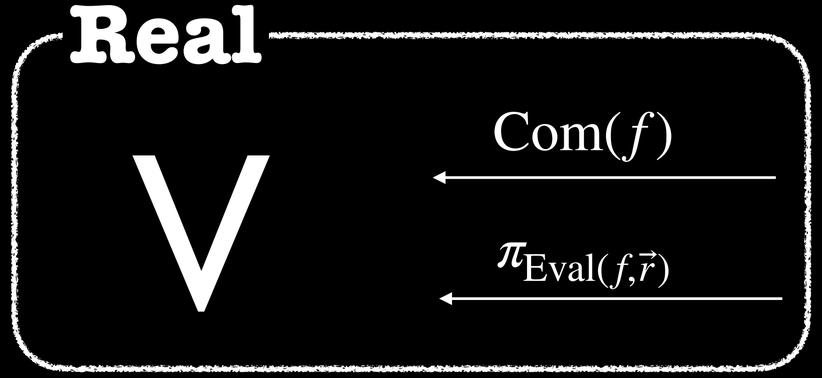
Probabilistic Interactive Oracle Proofs

Polynomial Commitment Scheme

Tensorcheck

At the end of inner product: $f(r_1, r_2, \ldots, r_n) = f^{(n)}$





Probabilistic Interactive Oracle Proofs

Univariate
Polynomial Commitment Scheme

Cryptographic compiler

Cryptographic compiler

An Elastic Polynomial Commitment Scheme

Streaming KZG

Overview

Streaming KZG

Overview

Setup

Bilinear group $(p, G, H, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$

$\mathsf{ck} = (G, \tau G, \tau^2 G, \ldots, \tau^n G)$,

$\mathsf{vk} = (H, \tau H)$

Streaming KZG

Overview

Setup

Bilinear group $(p, G, H, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$

$\mathsf{ck} = (G, \tau G, \tau^2 G, \ldots, \tau^n G)$,

$\mathsf{vk} = (H, \tau H)$

Commit

$$C := \mathsf{Com}_{\mathsf{ck}}(f) = f(\tau) \cdot G$$

Streaming KZG

Overview

Setup

Bilinear group $(p, G, H, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$

$\mathsf{ck} = (G, \tau G, \tau^2 G, \ldots, \tau^n G)$,

$\mathsf{vk} = (H, \tau H)$

Commit

$C := \mathsf{Com}_{\mathsf{ck}}(f) = f(\tau) \cdot G$

Stream of f:

f_N

Streaming KZG

Overview

Setup

Bilinear group $(p, G, H, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$

$\mathsf{ck} = (G, \tau G, \tau^2 G, \ldots, \tau^n G)$,

$\mathsf{vk} = (H, \tau H)$

Commit

$$C := \mathsf{Com}_{\mathsf{ck}}(f) = f(\tau) \cdot G$$

Stream of f:

f_N f_{N-1}

Streaming KZG

Overview

Setup

Bilinear group $(p, G, H, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$

$\mathsf{ck} = (G, \tau G, \tau^2 G, \ldots, \tau^n G)$,

$\mathsf{vk} = (H, \tau H)$

Commit

$$C := \mathsf{Com}_{\mathsf{ck}}(f) = f(\tau) \cdot G$$

Stream of f:

f_N f_{N-1} f_{N-2}

Streaming KZG

Overview

Setup

Bilinear group $(p, G, H, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$

$\mathsf{ck} = (G, \tau G, \tau^2 G, \ldots, \tau^n G)$,

$\mathsf{vk} = (H, \tau H)$

Commit

$C := \mathsf{Com}_{\mathsf{ck}}(f) = f(\tau) \cdot G$

Stream of f:

f_N f_{N-1} f_{N-2} f_{N-3}

Streaming KZG

Overview

Setup

Bilinear group $(p, G, H, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$

$\mathsf{ck} = (G, \tau G, \tau^2 G, \ldots, \tau^n G)$,

$\mathsf{vk} = (H, \tau H)$

Commit

$C := \mathsf{Com}_{\mathsf{ck}}(f) = f(\tau) \cdot G$

Stream of f:

f_N f_{N-1} f_{N-2} f_{N-3} f_{N-4}

Streaming KZG

Overview

Setup

Bilinear group $(p, G, H, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$

$\mathsf{ck} = (G, \tau G, \tau^2 G, \ldots, \tau^n G)$,

$\mathsf{vk} = (H, \tau H)$

Commit

$C := \mathsf{Com}_{\mathsf{ck}}(f) = f(\tau) \cdot G$

Stream of f:

f_N f_{N-1} f_{N-2} f_{N-3} f_{N-4} $\cdots$

Streaming KZG

Overview

Setup

Bilinear group $(p, G, H, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$

$\mathsf{ck} = (G, \tau G, \tau^2 G, \ldots, \tau^n G)$,

$\mathsf{vk} = (H, \tau H)$

Commit

$C := \mathsf{Com}_{\mathsf{ck}}(f) = f(\tau) \cdot G$

Stream of f:

f_N f_{N-1} f_{N-2} f_{N-3} f_{N-4} $\cdots$

Stream of ck:

ck_N ck_{N-1} ck_{N-2} ck_{N-3} ck_{N-4} $\cdots$

Streaming KZG

Overview

Setup

Bilinear group $(p, G, H, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$

$\mathsf{ck} = (G, \tau G, \tau^2 G, \ldots, \tau^n G)$,

$\mathsf{vk} = (H, \tau H)$

Commit

$C := \mathsf{Com}_{\mathsf{ck}}(f) = f(\tau) \cdot G$

Accumulate $f_i \cdot \mathsf{ck}_i$

Stream of f:

f_N f_{N-1} f_{N-2} f_{N-3} f_{N-4} $\cdots$

Stream of ck:

ck_N ck_{N-1} ck_{N-2} ck_{N-3} ck_{N-4} $\cdots$

Streaming KZG

Overview

Setup

Bilinear group $(p, G, H, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$

$\mathsf{ck} = (G, \tau G, \tau^2 G, \ldots, \tau^n G)$,

$\mathsf{vk} = (H, \tau H)$

Commit

$C := \mathsf{Com}_{\mathsf{ck}}(f) = f(\tau) \cdot G$

Accumulate $f_i \cdot \mathsf{ck}_i$

Stream of f:

f_{N-1} f_{N-2} f_{N-3} f_{N-4} $\cdots$

$C \mathrel{+}=$ f_N ck_N

Stream of ck:

ck_{N-1} ck_{N-2} ck_{N-3} ck_{N-4} $\cdots$

Streaming KZG

Overview

Setup

Bilinear group $(p, G, H, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$

$\mathsf{ck} = (G, \tau G, \tau^2 G, \ldots, \tau^n G)$,

$\mathsf{vk} = (H, \tau H)$

Commit

$C := \mathsf{Com}_{\mathsf{ck}}(f) = f(\tau) \cdot G$

Accumulate $f_i \cdot \mathsf{ck}_i$

Stream of f:

f_{N-1} f_{N-2} f_{N-3} f_{N-4} $\cdots$

$C \mathrel{+}=$

Stream of ck:

ck_{N-1} ck_{N-2} ck_{N-3} ck_{N-4} $\cdots$

Streaming KZG

Overview

Setup

Bilinear group $(p, G, H, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$

$\mathsf{ck} = (G, \tau G, \tau^2 G, \ldots, \tau^n G)$,

$\mathsf{vk} = (H, \tau H)$

Commit

$C := \mathsf{Com}_{\mathsf{ck}}(f) = f(\tau) \cdot G$

Accumulate $f_i \cdot \mathsf{ck}_i$

$C \mathrel{+}= f_{N-1} \quad \mathsf{ck}_{N-1}$

Stream of f:

f_{N-2} f_{N-3} f_{N-4} $\cdots$

Stream of ck:

ck_{N-2} ck_{N-3} ck_{N-4} $\cdots$

Streaming KZG

Overview

Setup

Bilinear group $(p, G, H, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$

$\mathsf{ck} = (G, \tau G, \tau^2 G, \ldots, \tau^n G)$,

$\mathsf{vk} = (H, \tau H)$

Eval

$\pi := \mathsf{Eval}_{\mathsf{ck}}(f, \alpha) = \mathsf{Com}_{\mathsf{ck}}(q)$

where $f = (x - \alpha)q(x) + r(x)$

Streaming KZG

Overview

Setup

Bilinear group $(p, G, H, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$

$\mathsf{ck} = (G, \tau G, \tau^2 G, \ldots, \tau^n G)$,

$\mathsf{vk} = (H, \tau H)$

Eval

$\pi := \mathsf{Eval}_{\mathsf{ck}}(f, \alpha) = \mathsf{Com}_{\mathsf{ck}}(q)$

where $f = (x - \alpha)q(x) + r(x)$

Ruffini's law: $q_i = f_{i+1} + q_{i+1}\alpha$!

Streaming KZG

Overview

Setup

Bilinear group $(p, G, H, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$

$\mathsf{ck} = (G, \tau G, \tau^2 G, \ldots, \tau^n G)$,

$\mathsf{vk} = (H, \tau H)$

Eval

$\pi := \mathsf{Eval}_{\mathsf{ck}}(f, \alpha) = \mathsf{Com}_{\mathsf{ck}}(q)$

where $f = (x - \alpha)q(x) + r(x)$

Ruffini's law: $q_i = f_{i+1} + q_{i+1}\alpha$!

Stream of f:

f_N f_{N-1} f_{N-2} f_{N-3} f_{N-4} ...

Streaming KZG

Overview

Setup

Bilinear group $(p, G, H, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$

$\mathsf{ck} = (G, \tau G, \tau^2 G, \ldots, \tau^n G)$,

$\mathsf{vk} = (H, \tau H)$

Eval

$\pi := \mathsf{Eval}_{\mathsf{ck}}(f, \alpha) = \mathsf{Com}_{\mathsf{ck}}(q)$

where $f = (x - \alpha)q(x) + r(x)$

Ruffini's law: $q_i = f_{i+1} + q_{i+1}\alpha$!

Stream of f :

f_N f_{N-1} f_{N-2} f_{N-3} f_{N-4} ...

Stream of q :

Streaming KZG

Overview

Setup

Bilinear group $(p, G, H, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$

$\mathsf{ck} = (G, \tau G, \tau^2 G, \ldots, \tau^n G)$,

$\mathsf{vk} = (H, \tau H)$

Eval

$\pi := \mathsf{Eval}_{\mathsf{ck}}(f, \alpha) = \mathsf{Com}_{\mathsf{ck}}(q)$

where $f = (x - \alpha)q(x) + r(x)$

Ruffini's law: $q_i = f_{i+1} + q_{i+1}\alpha$!

Stream of q :

q_N

Stream of f :

f_{N-1} f_{N-2} f_{N-3} f_{N-4} ...

Streaming KZG

Overview

Setup

Bilinear group $(p, G, H, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$

$\mathsf{ck} = (G, \tau G, \tau^2 G, \ldots, \tau^n G)$,

$\mathsf{vk} = (H, \tau H)$

Eval

$\pi := \mathsf{Eval}_{\mathsf{ck}}(f, \alpha) = \mathsf{Com}_{\mathsf{ck}}(q)$

where $f = (x - \alpha)q(x) + r(x)$

Ruffini's law: $q_i = f_{i+1} + q_{i+1}\alpha$!

Stream of f :

f_{N-2} f_{N-3} f_{N-4} ...

Stream of q :

q_N $q_{N-1} := q_N\alpha + f_{N-1}$

Streaming KZG

Overview

Setup

Bilinear group $(p, G, H, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$

$\mathsf{ck} = (G, \tau G, \tau^2 G, \ldots, \tau^n G)$,

$\mathsf{vk} = (H, \tau H)$

Eval

$\pi := \mathsf{Eval}_{\mathsf{ck}}(f, \alpha) = \mathsf{Com}_{\mathsf{ck}}(q)$

where $f = (x - \alpha)q(x) + r(x)$

Ruffini's law: $q_i = f_{i+1} + q_{i+1}\alpha$!

Stream of f :

f_{N-2} f_{N-3} f_{N-4} ...

Stream of q :

q_{N-1}

Streaming KZG

Overview

Setup

Bilinear group $(p, G, H, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$

$\mathsf{ck} = (G, \tau G, \tau^2 G, \ldots, \tau^n G)$,

$\mathsf{vk} = (H, \tau H)$

Eval

$\pi := \mathsf{Eval}_{\mathsf{ck}}(f, \alpha) = \mathsf{Com}_{\mathsf{ck}}(q)$

where $f = (x - \alpha)q(x) + r(x)$

Ruffini's law: $q_i = f_{i+1} + q_{i+1}\alpha$!

Stream of q :

q_{N-1}

Stream of f :

f_{N-2} f_{N-3} f_{N-4} ...

Coding elastic SNARKs

Implementing elastic arguments

Gemini

Implementing elastic arguments

Gemini

Polynomial Commitment schemes

Tensorcheck

Elastic KZG

Elastic PST

Implementing elastic arguments

Gemini

Elastic Inner-Product

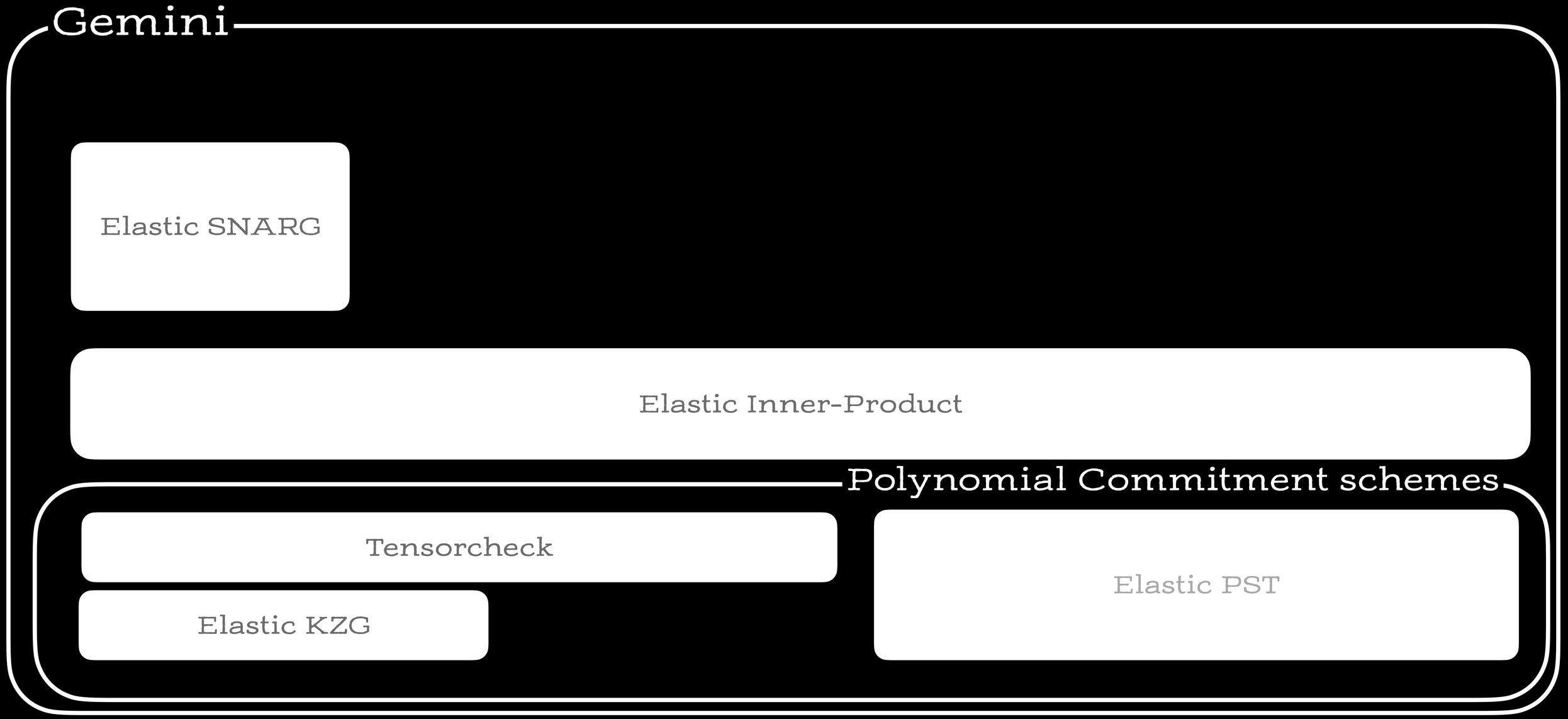
Polynomial Commitment schemes

Tensorcheck

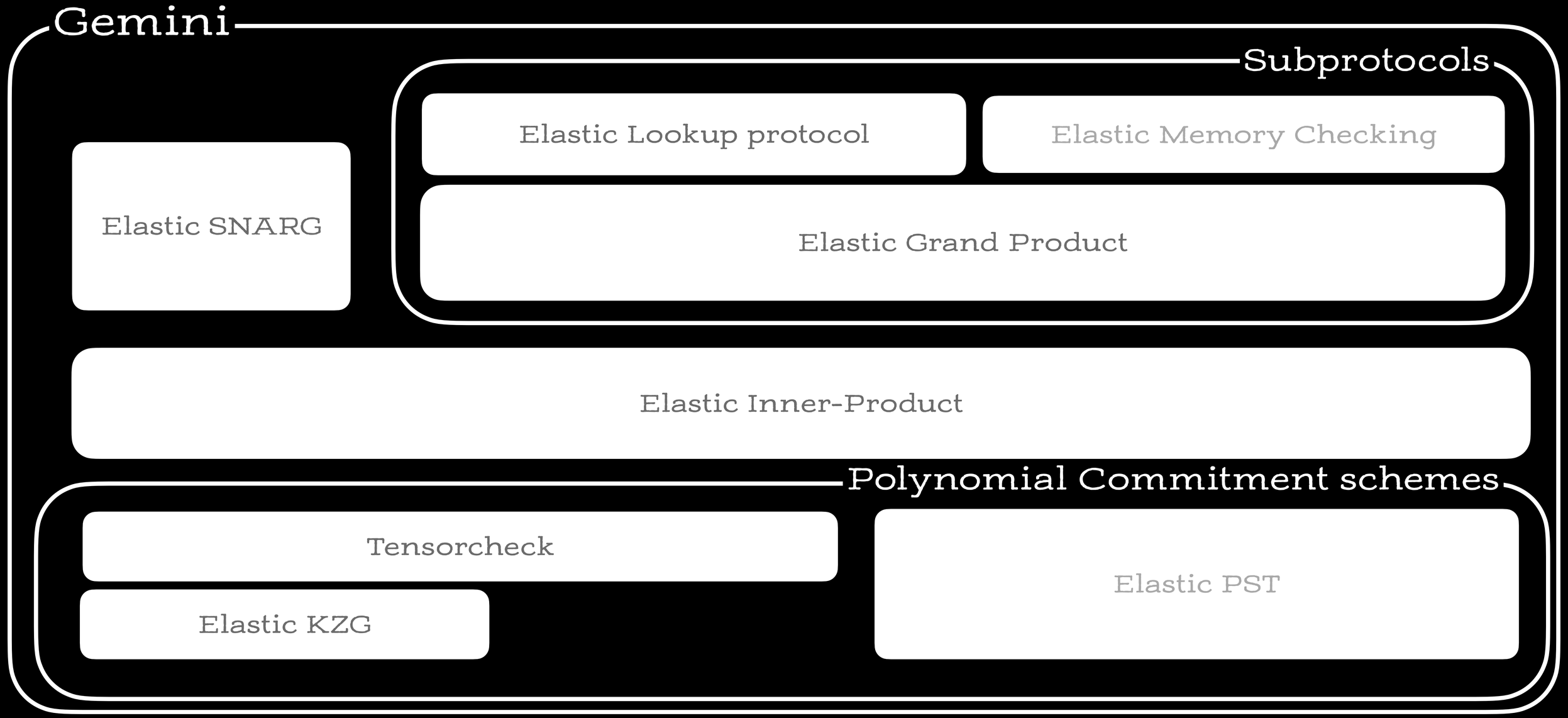
Elastic KZG

Elastic PST

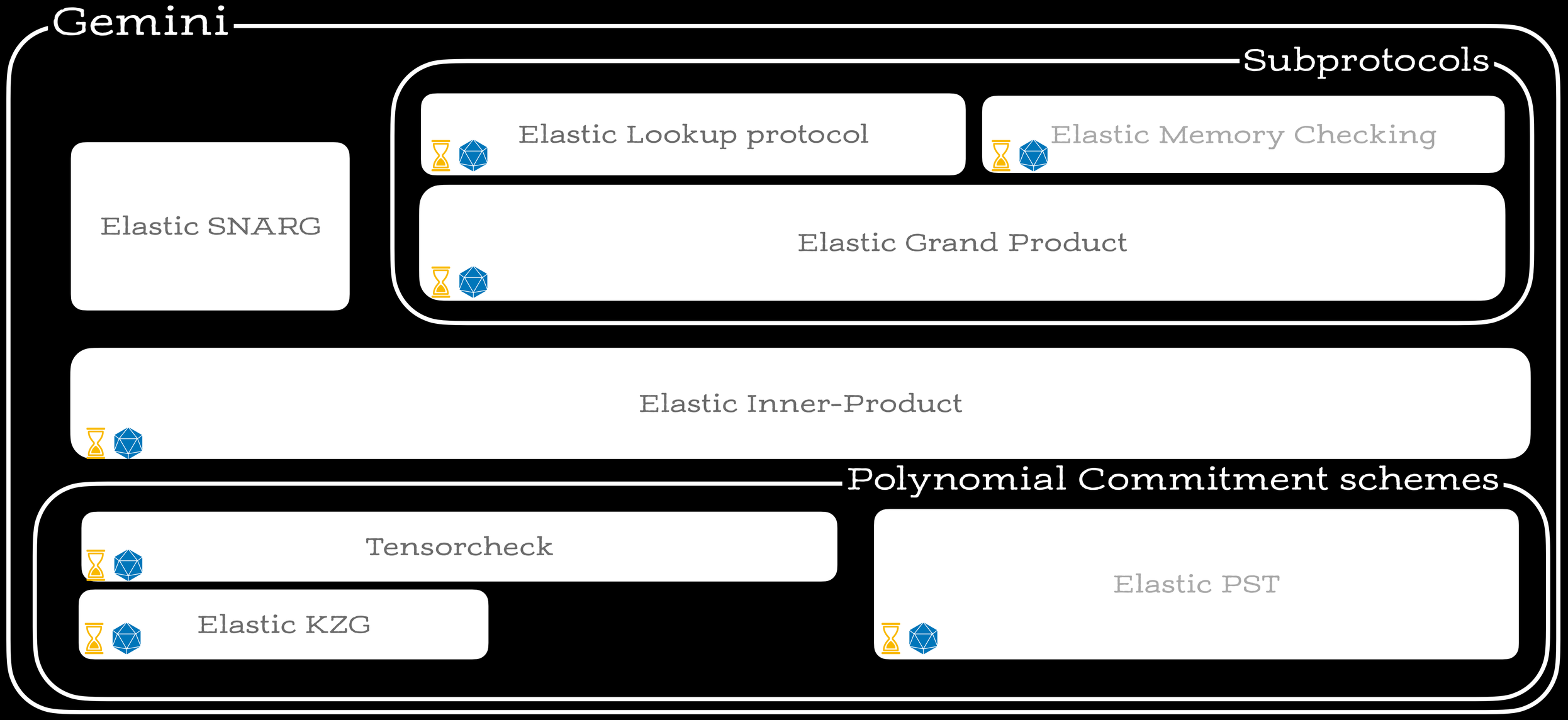
Implementing elastic arguments



Implementing elastic arguments



Implementing elastic arguments



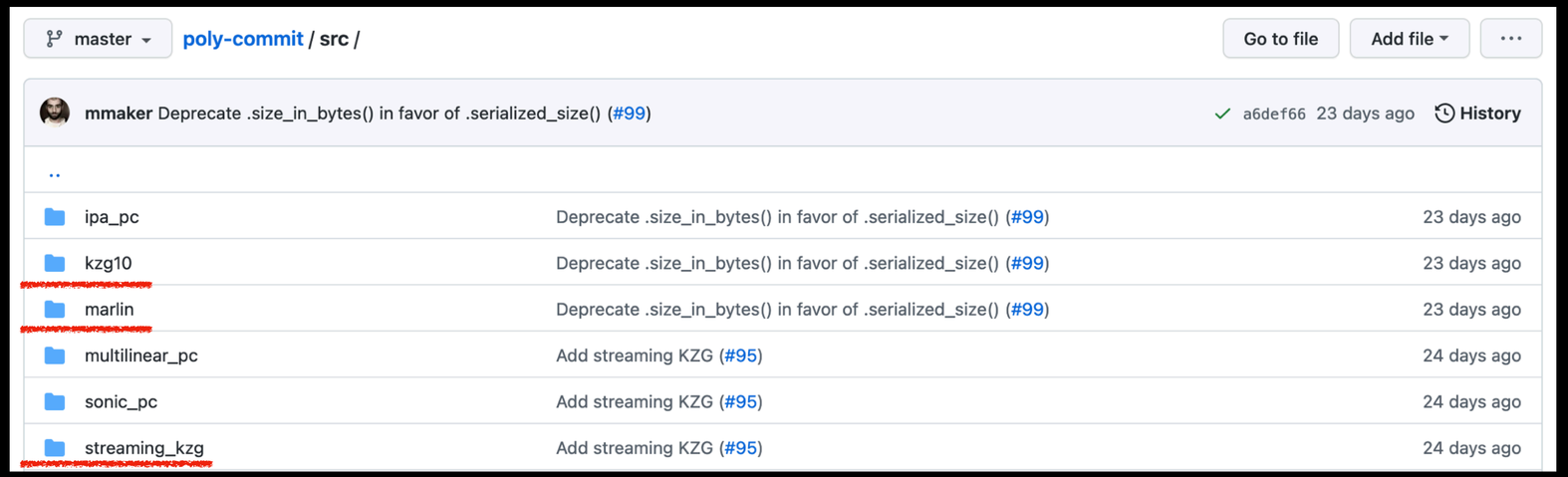
Implementation overview

Contributing to arkworks-rs: streaming KZG and PST.

Streams are Iterators: embed computation tree in the type.

Implementation overview

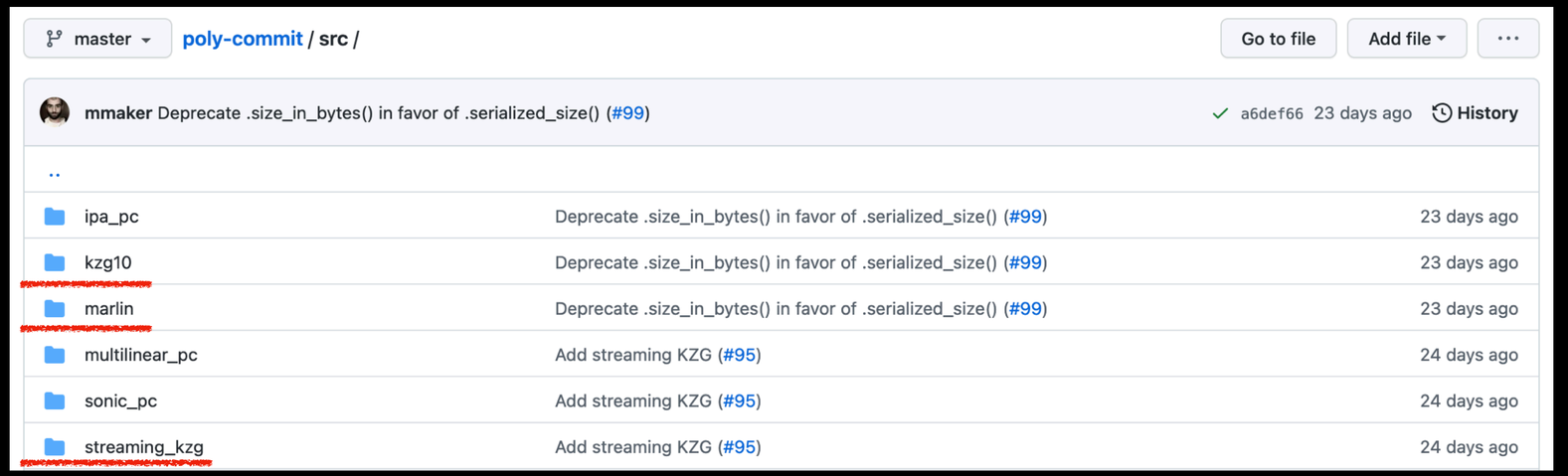
Contributing to arkworks-rs: streaming KZG and PST.



Streams are Iterators: embed computation tree in the type.

Implementation overview

Contributing to arkworks-rs: streaming KZG and PST.



master ▾ poly-commit / src / Go to file Add file ▾ ···

mmaker Deprecate .size_in_bytes() in favor of .serialized_size() (#99) ✓ a6def66 23 days ago History

..		
ipa_pc	Deprecate .size_in_bytes() in favor of .serialized_size() (#99)	23 days ago
kzg10	Deprecate .size_in_bytes() in favor of .serialized_size() (#99)	23 days ago
marlin	Deprecate .size_in_bytes() in favor of .serialized_size() (#99)	23 days ago
multilinear_pc	Add streaming KZG (#95)	24 days ago
sonic_pc	Add streaming KZG (#95)	24 days ago
streaming_kzg	Add streaming KZG (#95)	24 days ago

Streams are Iterators: embed computation tree in the type.

```
SortedStreamer<'_, _,
      AlgebraicHash<'_, <E as PairingEngine>::Fr,
                Tensor<'_, <E as PairingEngine>::Fr>, IterableRange>,
                      JointRowStream<'_, SM, SM, SM, <E as PairingEngine>::Fr>
                >
      >
>
```

Benchmark huge circuits

Gemini in practice

Previous works: benchmarking stops at millions of gates.

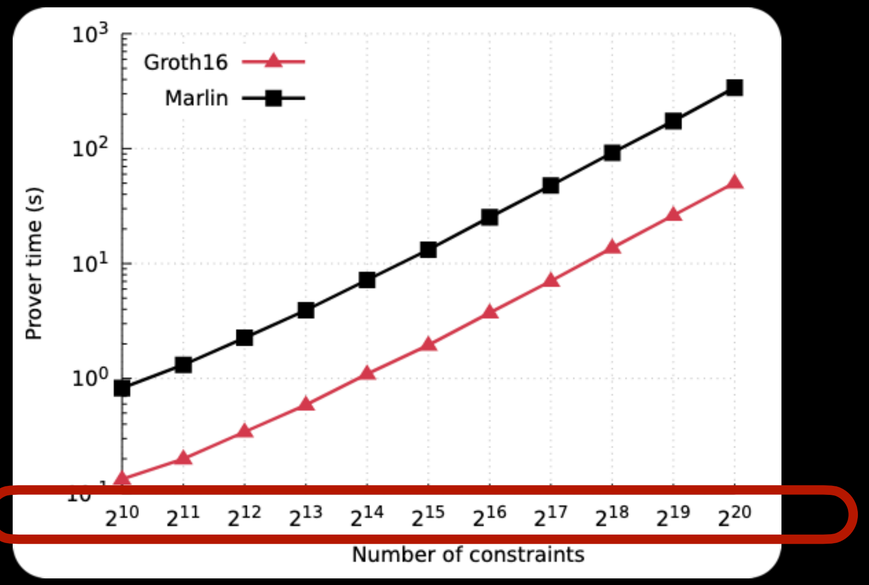
Gemini in practice

Previous works: benchmarking stops at millions of gates.

	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}
Groth16	0.17	0.26	0.46	0.85	1.5	2.8	5.4	10.1	23.2	44.7	76.2
Hyrax	1.2	1.7	2.8	4.1	7.2	13.9	22.5	44.6	90	181	447
Ligero	0.2	0.3	0.6	1.2	2.3	3.4	6.7	13.7	27.2	56.3	112
Ligero-heuristic	0.16	0.3	0.6	1.3	2.4	3.5	6.6	13	25	51.3	101
Aurora	0.7	1.2	2.5	4.6	9	17.3	36	69	140	282	688
Aurora-heuristic	0.4	0.7	1.4	3.2	5.8	12.2	24.8	52.2	108	224	509
Fractal	1	1.8	3.6	6.7	15	30	61	125	337	–	–
Fractal-heuristic	0.8	1.5	3	6.5	14	29	60	125	342	–	–
SpartanNIZK	0.02	0.03	0.04	0.06	0.1	0.17	0.33	0.57	1.1	2.14	4.5
SpartanSNARK	0.07	0.13	0.21	0.39	0.79	1.3	2.6	4.9	9.2	18.5	36.3

FIGURE 7—Prover's performance (in seconds) for varying R1CS instance sizes under different schemes.

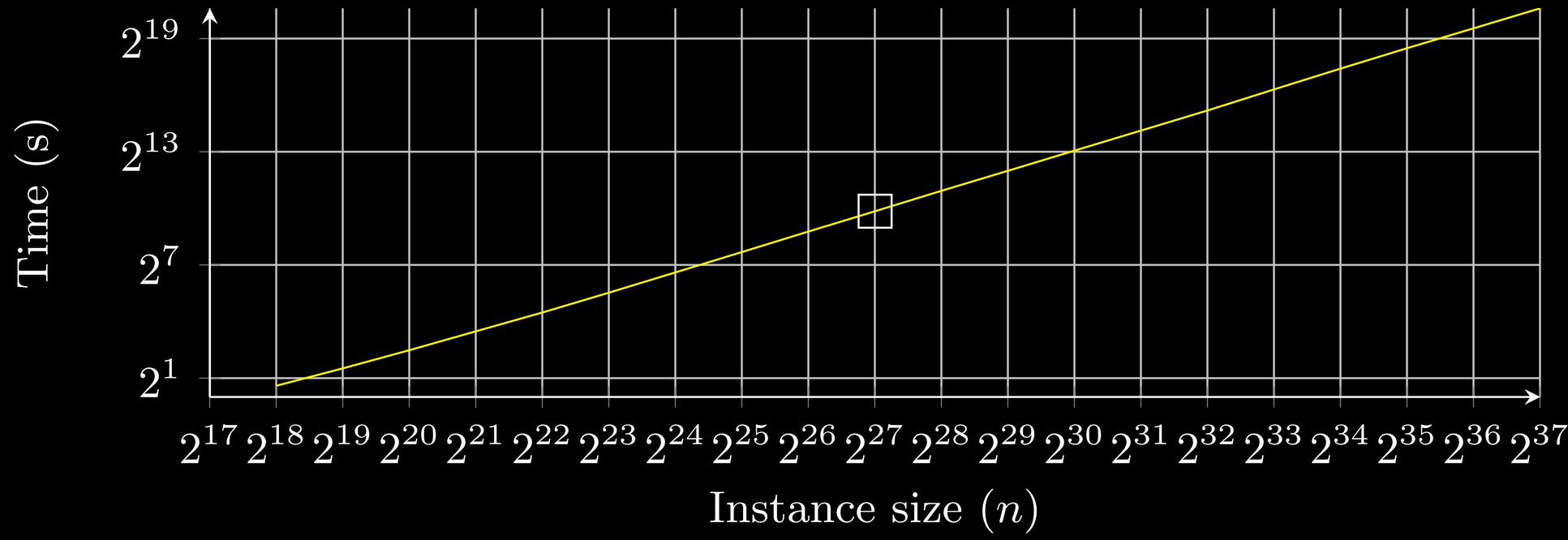
Srinath Setty, Spartan: Efficient and general-purpose zkSNARKs without trusted setup, 2019



Chiesa, Maller, Hu, Mishra, Vesely, Ward: Marlin: Preprocessing SNARKs with universal updatable CRS

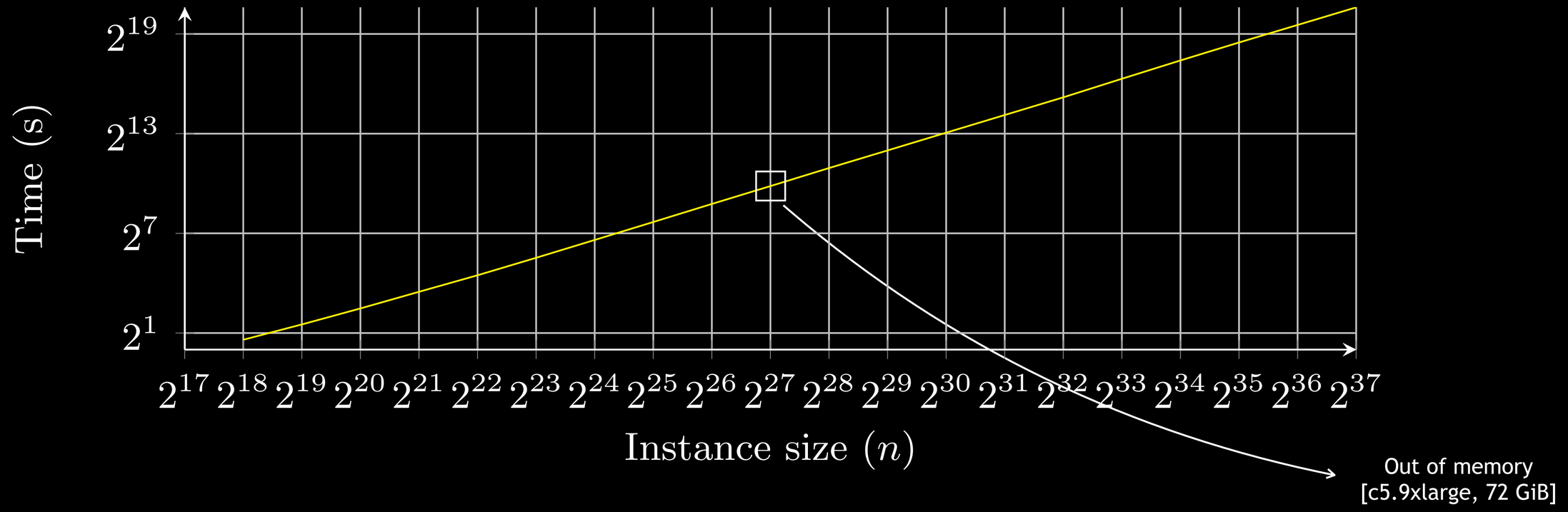
Gemini in practice: SNARK

Hundreds of billions of gates



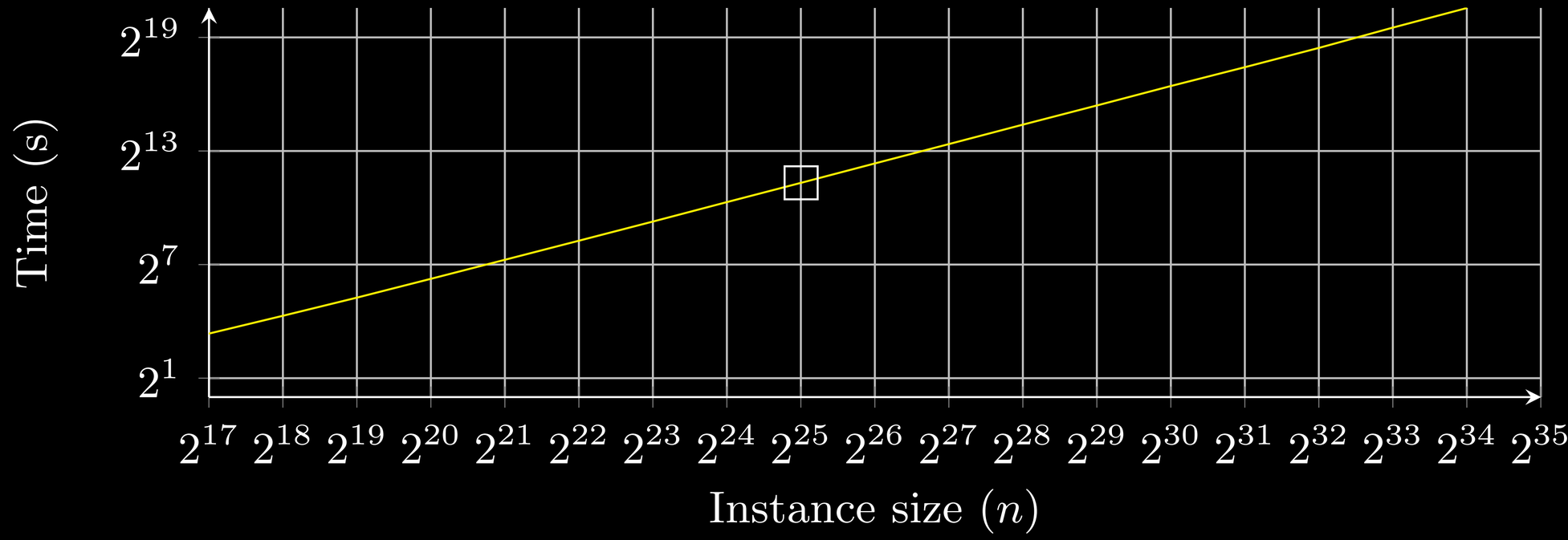
Gemini in practice: SNARK

Hundreds of billions of gates



Gemini in practice: preprocessing SNARK

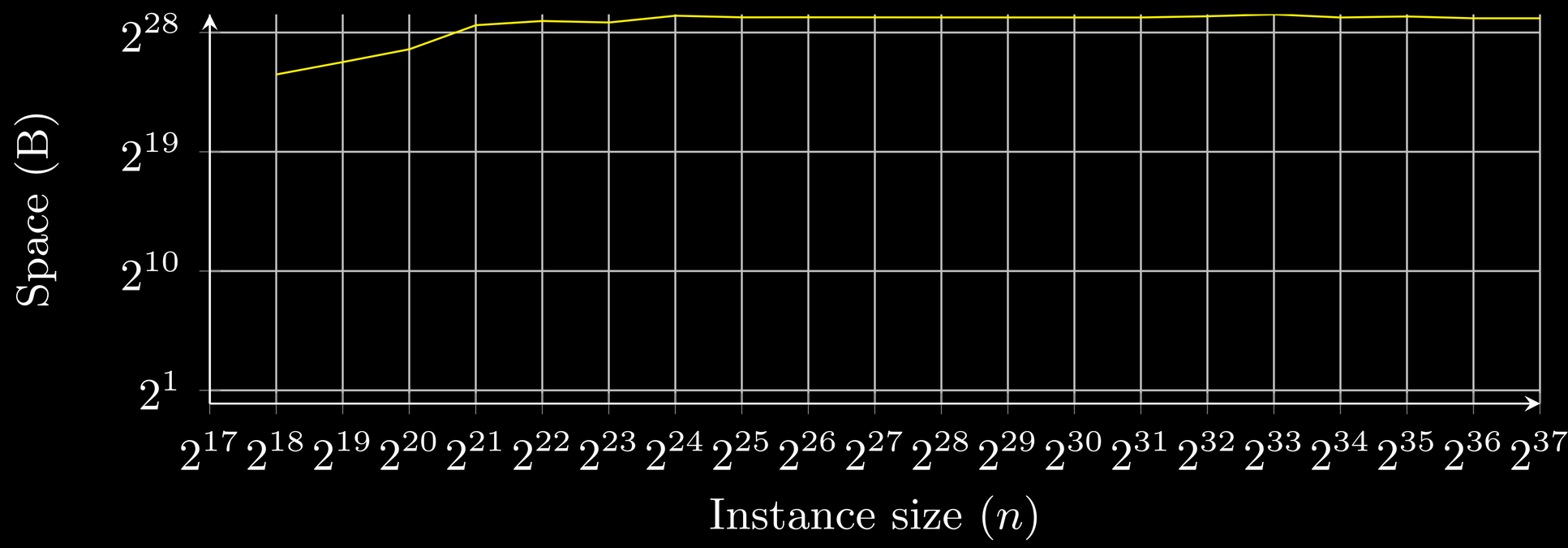
Tens of billions of gates



Proving time: $O(n \log^2 n)$

Gemini in practice: SNARK

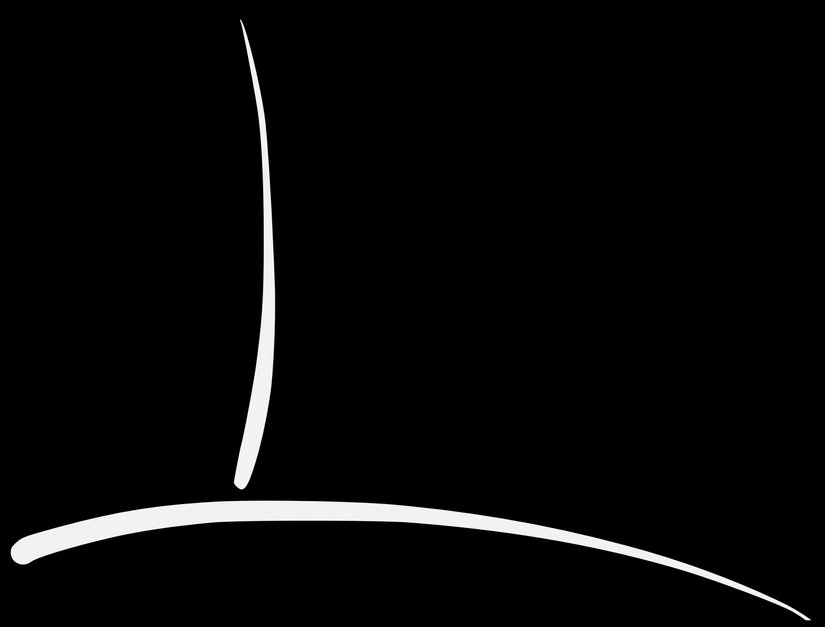
If you care about logarithmic factors, you care about constants



Proving space: $O(\log n)$

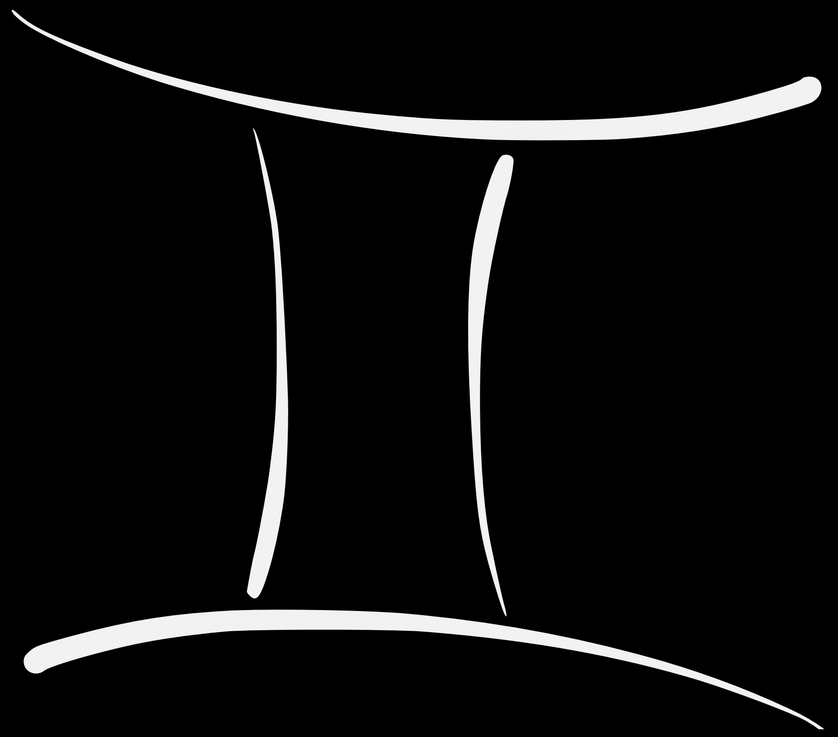
ia.cr/2022/420

arkworks.rs/gemini



ia.cr/2022/420

arkworks.rs/gemini



ia.cr/2022/420

arkworks.rs/gemini