Efficient Batch Zero-Knowledge Arguments for Low Degree Polynomials

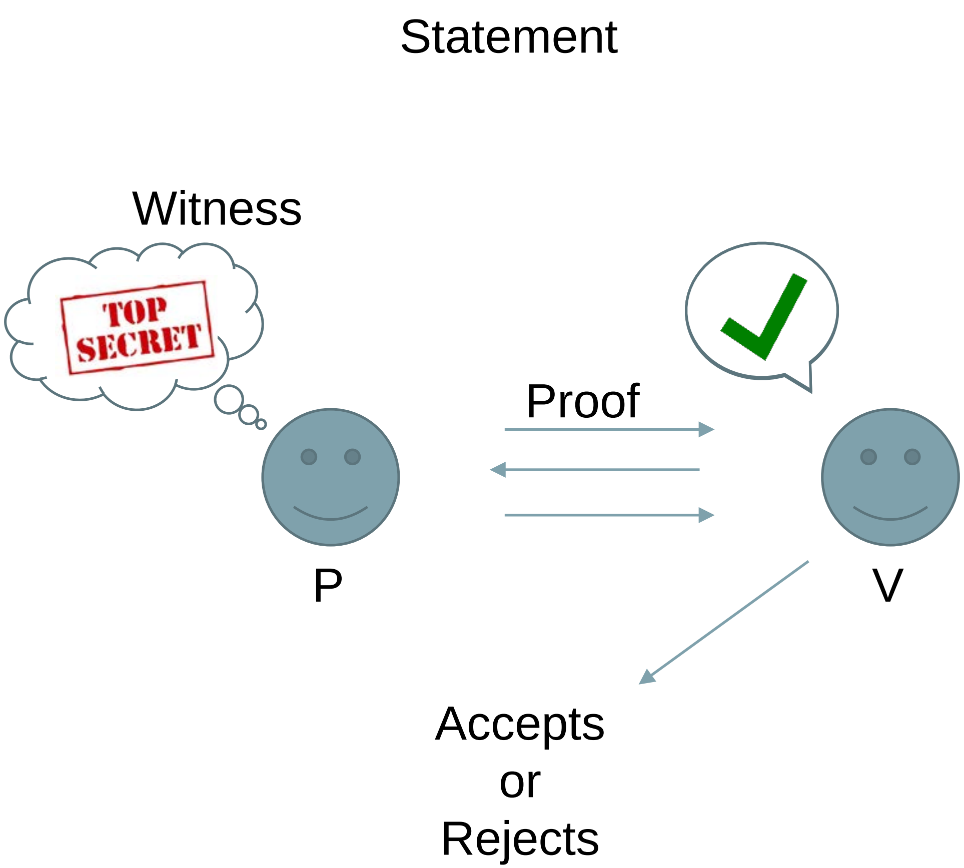
Jonathan Bootle, Jens Groth
University College London

European Research Council
Established by the European Commission

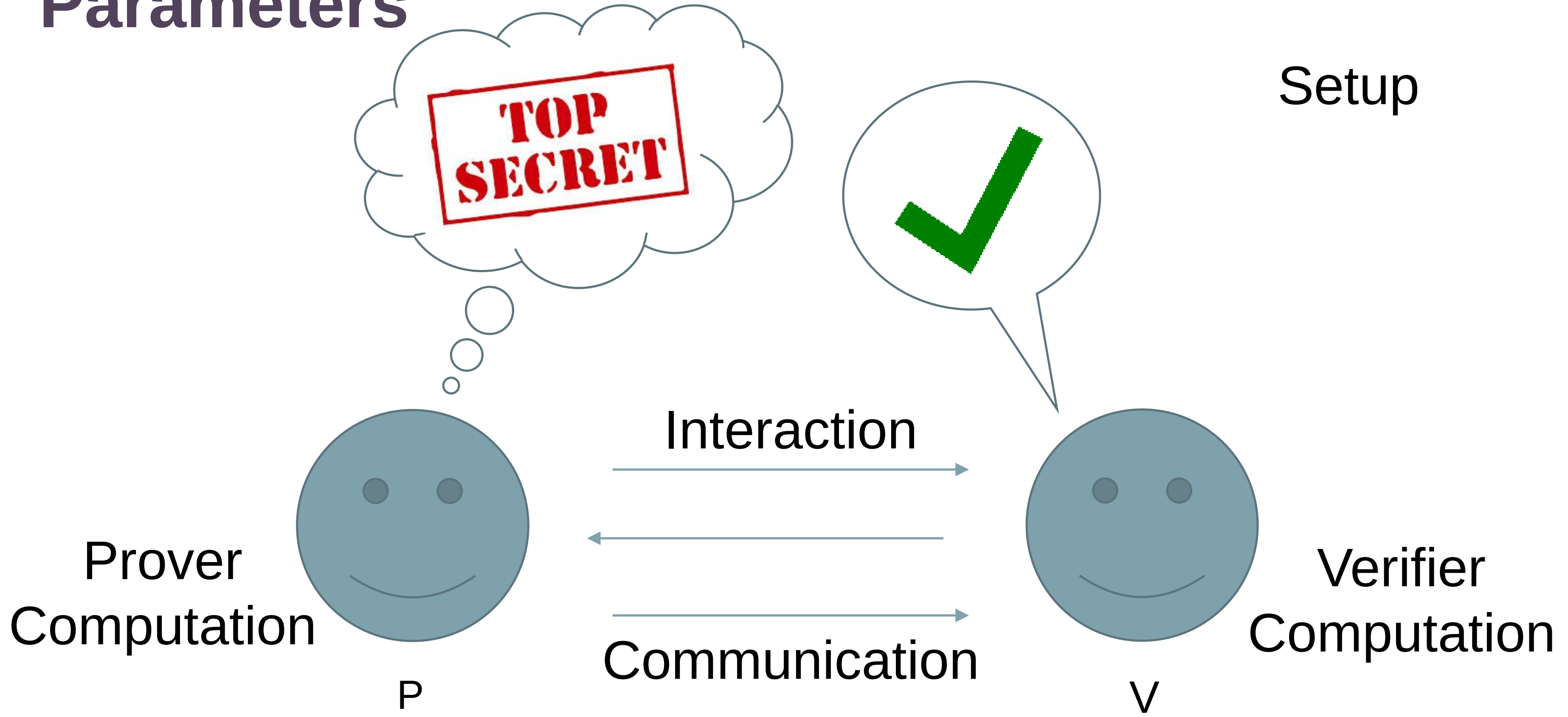
Zero Knowledge Proofs

- Completeness
- Soundness
- Zero-Knowledge

- Proof of Knowledge
- Interactive
- Public-coin



Parameters



Goal: reduce costs for special statements

This Work

$\lambda \in \{\lambda_1, \dots \lambda_N\}$

- Efficient for low-depth circuits like set membership and polynomial evaluation $v = h_0 + h_1 u + h_2 u^2 + \ \dots + h_N u^N$
- Works with homomorphic commitments

- Generalises and explains some previous works
- Avoids reductions to general statements

Applications

- Set membership proofs

$$\lambda \in \{\lambda_1, \ldots \lambda_N\}$$

-$O(\log N)$ communication, improved constants.
-Tune to get protocol with $O(1)$ commitments

- Polynomial evaluation

$$v = h_0 + h_1 u + h_2 u^2 + \ \ldots + h_n u^N$$

-$O(\log N / \log\log N)$ communication, asymptotic improvement.

Statement

- Secret witness $\boldsymbol{a}$
- Public polynomials $\boldsymbol{P}$ and $\boldsymbol{Q}$

$$\boldsymbol{P}(\boldsymbol{a}) = 0$$

- Commitment c

$$c = com(\boldsymbol{Q}(\boldsymbol{a}); r)$$

Example: committed bit

- $P(a) = a(1 - a)$
- $Q(a) = a$

- Proof that $c = com(a; r)$ and $a \in \{0,1\}$

Tweaking Statements

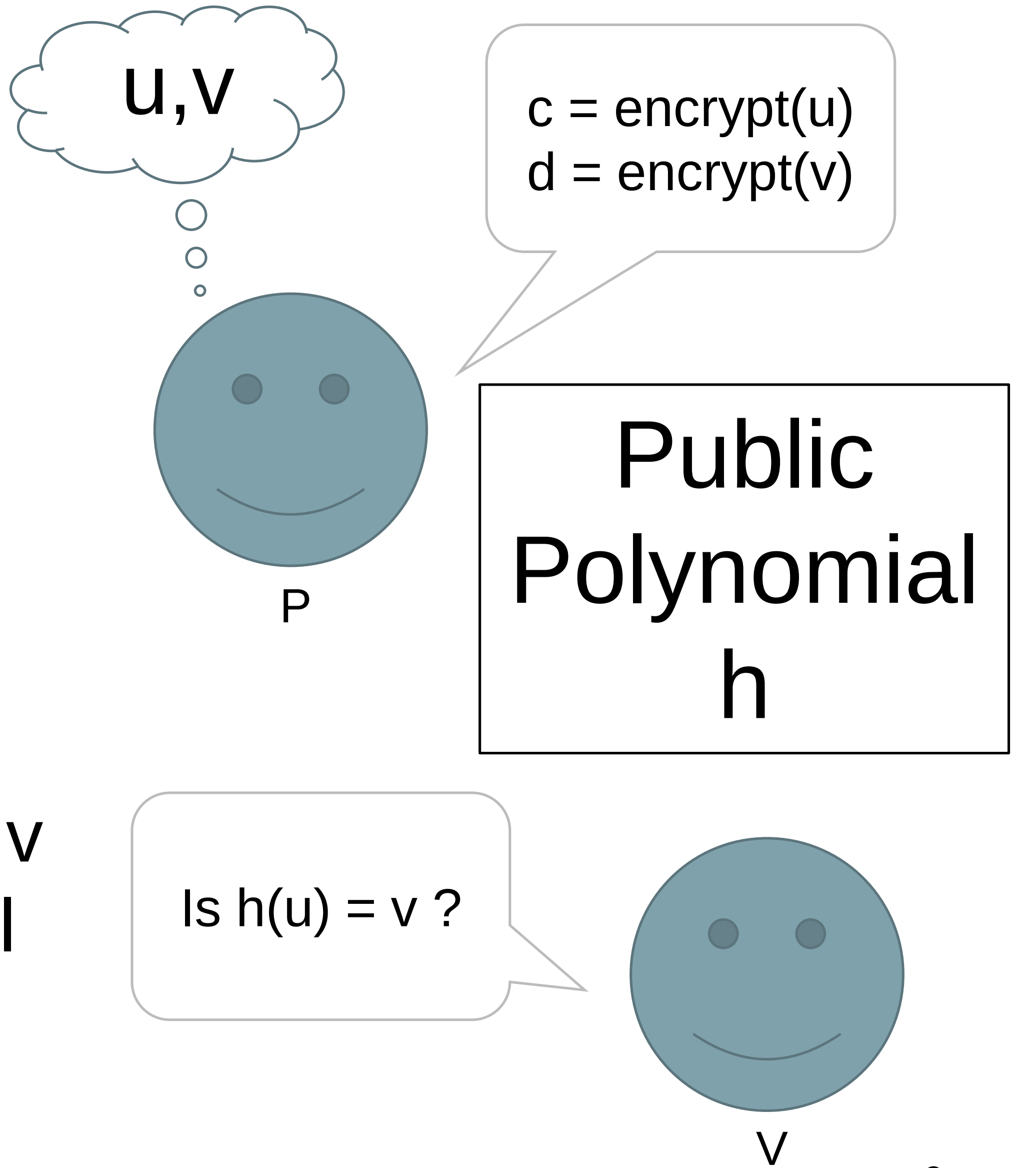
- Secret witness $\boldsymbol{a}$
- Public tweak $\boldsymbol{b}$
- Public polynomials $\boldsymbol{P}$ and $\boldsymbol{Q}$
- Commitment c

$$\boldsymbol{P}(\boldsymbol{a}, \boldsymbol{b}) = 0$$

$$c = com(\boldsymbol{Q}(\boldsymbol{a}, \boldsymbol{b}); r)$$

Polynomial Evaluation Proofs

- The prover has secret numbers, u and v
- The prover convinces the verifier that h(u) = v for a public polynomial h



Polynomial Evaluation Proofs

- $\boldsymbol{a} = \left(1, u, u^2, u^3, \ldots, u^{\mathrm{deg}(h)}\right)$
- $\boldsymbol{P}(\boldsymbol{a}) = a_2 \cdot \left(a_1, a_2, \ldots, a_{\mathrm{deg}(h)-1}\right) - \left(a_2, \ldots, a_{\mathrm{deg}(h)}\right)$

- $\boldsymbol{b} =$ coefficients of h
- $Q(\boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{a} \cdot \boldsymbol{b} = h(u)$

Batch Proofs

- Secret witnesses $\boldsymbol{a_{1,1}, \ldots a_{m,n}}$
- Public tweak $\boldsymbol{b}$
- Public polynomials $\boldsymbol{P}$ and $\boldsymbol{Q}$
- Commitments $c_1, \ldots c_m$

$\boldsymbol{P}(\boldsymbol{a_{i,j}}, \boldsymbol{b}) = 0$ for all i, j

For all i, $c_i = com(\boldsymbol{Q}(\boldsymbol{a_{i,1}}, \boldsymbol{b}), \ldots, \boldsymbol{Q}(\boldsymbol{a_{i,n}}, \boldsymbol{b}); r_i)$

t instances $\rightarrow \sqrt{t}$ communication overhead

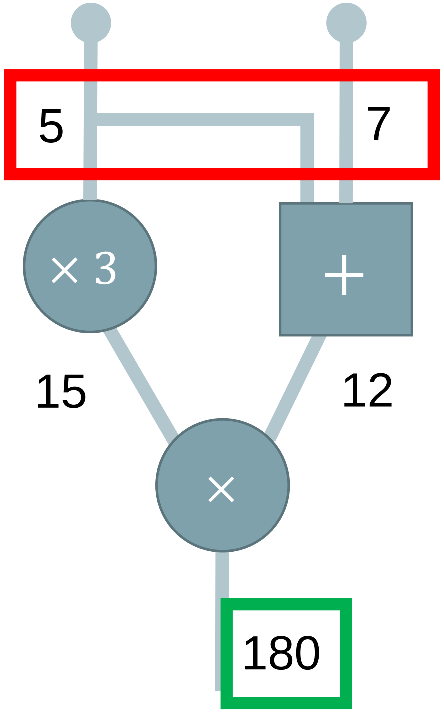
Polynomial Evaluation Proofs

Previous work	Proof Size	Year
Brands et al	$O\left(\sqrt{N}\right)$	2007
Bayer, Groth	$O(\log N)$	2013
This Work	$\frac{5 \log N}{\log \log N}$	2018

N = degree of polynomial

Arithmetic Circuits

- N multiplication gates
- Prover knows inputs
- Publicly known outputs
- Check inputs give the correct outputs



General Arithmetic Circuit Proofs

Previous work	Proof Size	Year
Cramer, Damgård	$O(N)$	1997
Groth	$O(\sqrt{N})$	2009
BCCGP	$6 \log N$	2016
Bulletproofs	$2 \log N$	2017

N = degree of polynomial

Is this always the best possible?

General Arithmetic Circuit Proofs

Previous work	Proof Size	Year
Cramer, Damgård	$6N$	1997
Groth	$6\sqrt{N}$	2009
BCCGP	$6\log N$	2016
Bulletproofs	$2\log N$	2017
This Work	$\frac{5\log N}{\log\log N}$	2018

Overview

Statement

Polynomials

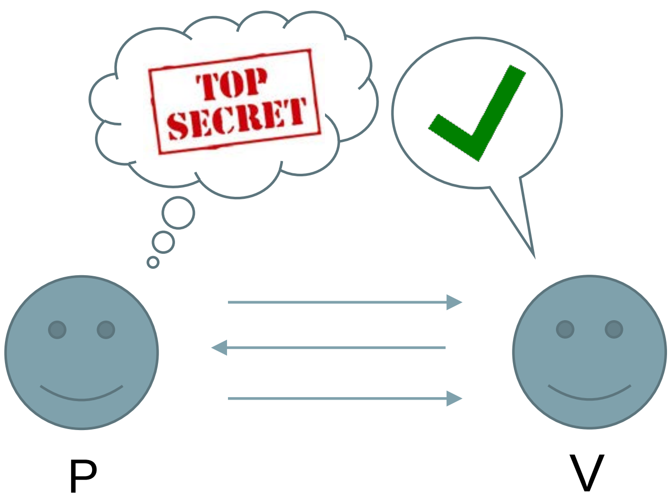
Polynomial Commitments

Protocol

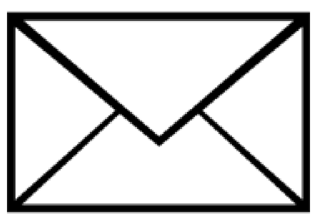
$$\boldsymbol{P}(\boldsymbol{a},\boldsymbol{b}) = 0, c = com(\boldsymbol{Q}(\boldsymbol{a},\boldsymbol{b}); r)$$

$$\boldsymbol{P}(\boldsymbol{a}x + \boldsymbol{a}', \boldsymbol{b}) = \sum_i \boldsymbol{p}_i^{x^i}$$

PolyCommit, PolyEval, PolyVerify



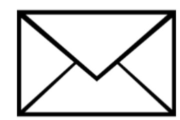
Commitments



Pedersen Commitments

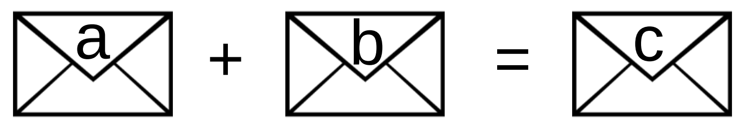
- Compressing

2	6	6	2	0	1	9	2	7	4



- Homomorphic

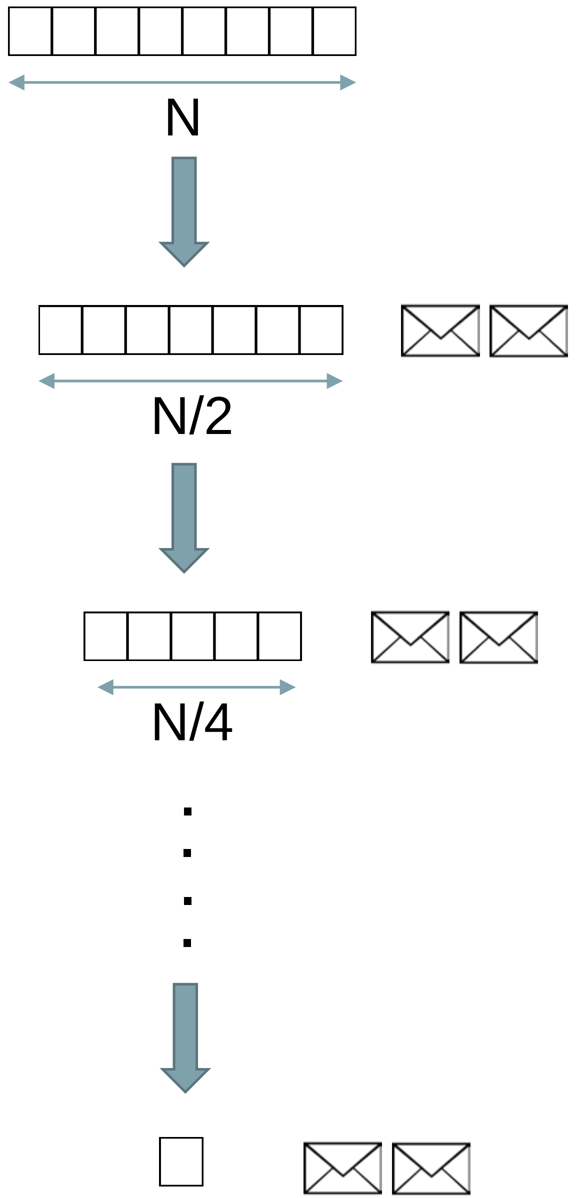
a + b = c



- $C = com(m_1, \ldots, m_k; r) = g_1^{m_1} g_2^{m_2} \ldots g_k^{m_k} h^{r}$
- $g_1, g_2, \ldots, g_k, h$ random elements of G

Previous Arguments

- Reduce AC-SAT to a scalar-product check
- Compress vectors of length N
- O(log N) communication costs



This Work

$$\boldsymbol{a} = \left(1, u, u^2, u^3, \dots, u^{\deg(h)}\right)$$
$$\boldsymbol{P}(\boldsymbol{a}) = a_2 \cdot \left(a_1, a_2, \dots, a_{\deg(h)-1}\right) - \left(a_2, \dots, a_{\deg(h)}\right)$$

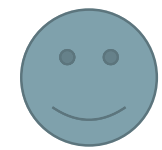
- Works directly with circuits
- Degree and number of inputs determines performance
- Competitive performance for circuits of degree $O(\log N)$ and $O(\log N)$ inputs, but $O(N)$ gates

Proof of Knowledge

$$c = com(a; r)$$

$$P(a) = 0$$
$$Q(a) = a$$

P



V

Choose random $a', s \in \mathbb{Z}_p$
Compute $B = com(a'; s)$

$\xrightarrow{\quad B \quad}$

$\xleftarrow{\quad x \quad}$

Choose random $x \in \mathbb{Z}_p$

$f = ax + a'$
$z = rx + s$

$\xrightarrow{\quad f, z \quad}$

Check that
$com(f; z) = C^x B$

Committed Bits

- A bit a satisfies $a(1-a)=0$
- Prover sends $f=ax+a'$ to the verifier
- Embed $a(1-a)$ into some polynomial

- $f(x-f) = a(1-a)x^2 + k_1x + k_2$ for some constants k_1, k_2

- The prover commits to k_1, k_2 beforehand and shows that $f(x-f)$ is actually a polynomial of degree 1

Committed Bits Protocol

P

$$a \in \{0,1\} \qquad P(a) = a(1-a)$$
$$c = com(a;r) \qquad Q(a) = a$$

V

- Commit to a'
- $f = aX + a'$
- $P(f) = k_1 X + k_2$

Compute $K_1 = com(k_1; t_1)$
Compute $K_2 = com(k_2; t_2)$

commitments $\longrightarrow$

$\xleftarrow{x}$ Choose random $x \in \mathbb{Z}_p$

$f = ax + a'$
$z = rx + s$
$v = t_1 x + t_2$

$\xrightarrow{f, z, v}$

Check that
$com(f; z) = C^x B$
$com(f(x-f); v) = K_1^x K_2$

Committed Bits Protocol

P

$a \in \{0,1\}$ $P(a) = a(1-a)$

$c = com(a;r)$ $Q(a) = a$

V

Commit to random values to hide witness

commitments

Compute coefficients of a polynomial. Commit to them.

$\xrightarrow{\hspace{3em}}$

$\xleftarrow{\;x\;}$ Choose random $x \in \mathbb{Z}_p$

Send witness, which is hidden by adding random values.

$\xrightarrow{f, z, v}$

Check witness.
Check polynomial has degree 1, not 2.

General Recipe

- Commit to secret vector $\boldsymbol{a}$ and random vector $\boldsymbol{a}'$
- Prover sends $\boldsymbol{f} = \boldsymbol{a}x + \boldsymbol{a}'$ to the verifier

- If $\boldsymbol{P}(\boldsymbol{a}) = \boldsymbol{0}$, then $\boldsymbol{P}(\boldsymbol{f})$ has a zero in the leading x coefficient
- Commit to other coefficients in advance

General Recipe

- $P(\boldsymbol{f}, \boldsymbol{b}) = P(\boldsymbol{a}, \boldsymbol{b}) + \sum_i \boldsymbol{p}_i x^i$
- Make commitments $P_i = com(\boldsymbol{p}_i)$
- Check com
- Also want t
- $Q(\boldsymbol{f}, \boldsymbol{b}) = Q(\boldsymbol{a}, \boldsymbol{b}) + \sum_i \boldsymbol{q}_i x^i$
- Make commitments $Q_i = com(\boldsymbol{q}_i)$
- Check $com(Q(\boldsymbol{f}, \boldsymbol{b}); v) = c \cdot \prod_i Q_i^{x^i}$

Improve efficiency using polynomial commitment protocol

General Protocol

$$P(a,b) = 0$$
$$c = com(Q(a,b);r)$$

P

V

Commit to random values to hide witness

commitments →

Compute coefficients of a polynomial. Commit to them. PolyCommit

← x

Choose random $x \in \mathbb{Z}_p$

f, z, v →

Send witness, which is hidden by adding random values. PolyEval

Tuneable parameters

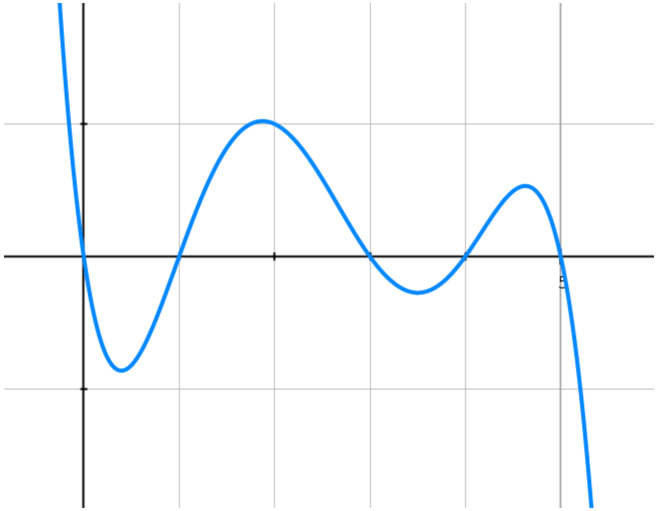
Check witness.
Check polynomials.
PolyVerify

Batch Protocol Idea

- Verify many instances of the same relation in parallel using interpolation
- Evaluate polynomials on single elements

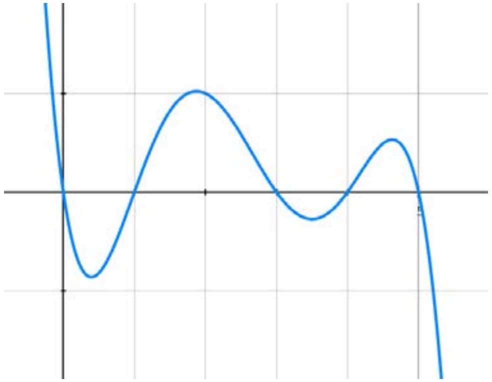
Lagrange Polynomials

- Interpolation points $z_1, \dots, z_n$
- $l_i(X) = \prod_{j \neq i} \frac{X - z_j}{z_i - z_j}$ for $1 \leq i \leq n$
- $l_i(z_j) = \begin{cases} 1 \text{ if } i = j \\ 0 \text{ otherwise} \end{cases}$



- Set $l_0(X) = \prod_i (X - z_j)$

Batched Protocol



- Secret vectors $\boldsymbol{a}_{1,1}, \dots \boldsymbol{a}_{m,n}$
- Prover sends $\boldsymbol{f}_i = \sum_j \boldsymbol{a}_{i,j}\, l_j(x) + \boldsymbol{a}' l_0(x)$ to the verifier

- If $P\left(\boldsymbol{a}_{i,j}\right) = 0$, then $P(\boldsymbol{f}_i) = 0\ mod\ l_0(x)$
- Use polynomial commitment

Advantages over General Circuit Protocols

- Same or better communication complexity for low-depth circuits

- $O(N) \rightarrow O(\log N)$ cryptographic operations
- $O(\log N) \rightarrow 3$ round protocols

- No special properties needed e.g. key homomorphic commitments

Thanks!