polynomial commitments

building block for universal SNARKs

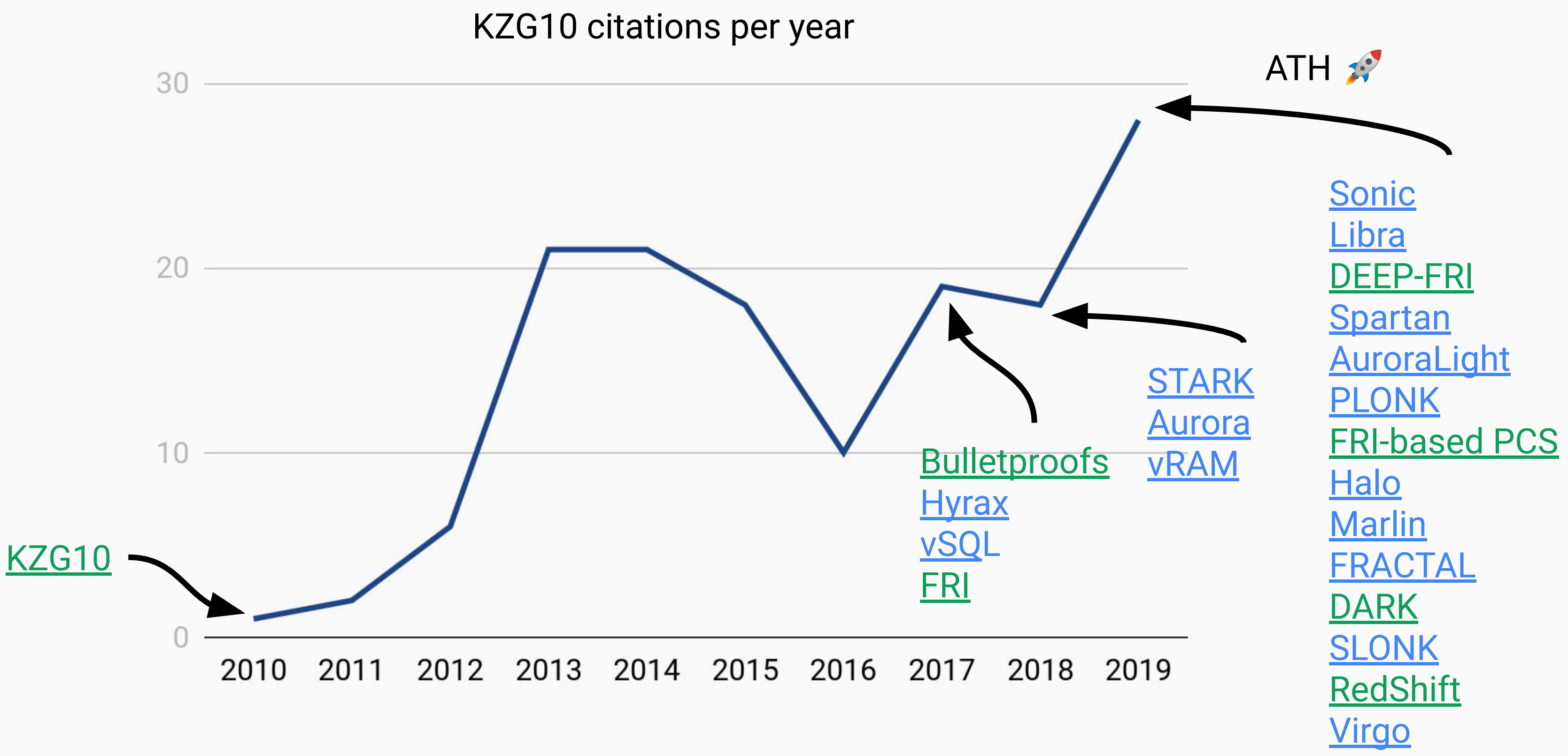
part 1—context

part 2—landscape

part 3—mechanics

part 4—gadgets

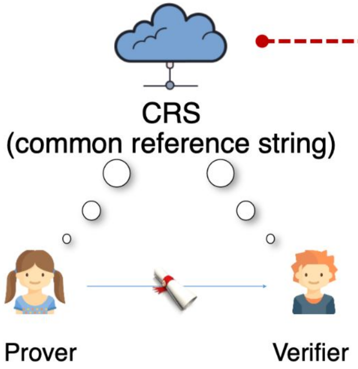
historical perspective



Zk-SNARK의 종류

매번 셋업을 다시해줘야함

Trusted Setup



Pinocchio, [Gro16], [GM17],
[Lip19], [KLO19]...

Universal

Transparent

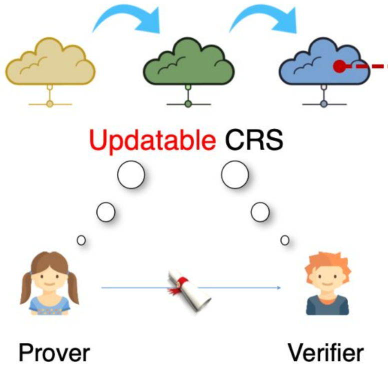
믿을 만한 제 3자가 필요

- 공용키 제공 주체의 비밀정보 누설 시 증명자는 실제 맞지 않는 연산값으로 증명을 만들어내는 문제가 발생함

Zk-SNARK의 종류

하나의 큰 셋업을해서 업데이트같이해서 나만에 셋업을해서 사용

Trusted Setup | **Universal** | **Transparent**



만들어진 공용키 업데이트

- 공용키로 특정 코드에 적합한 새로운 키 생성 가능

유연성, 자유도, 적용성 높음

Zk-SNARK의 종류

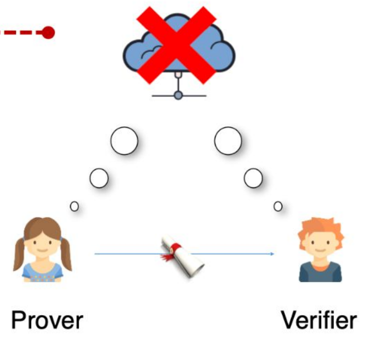
셋업이 필요없다

Trusted Setup | **Universal** | **Transparent**

공용키가 존재하지 않음

- 증명을 위해서는 해시함수만 필요

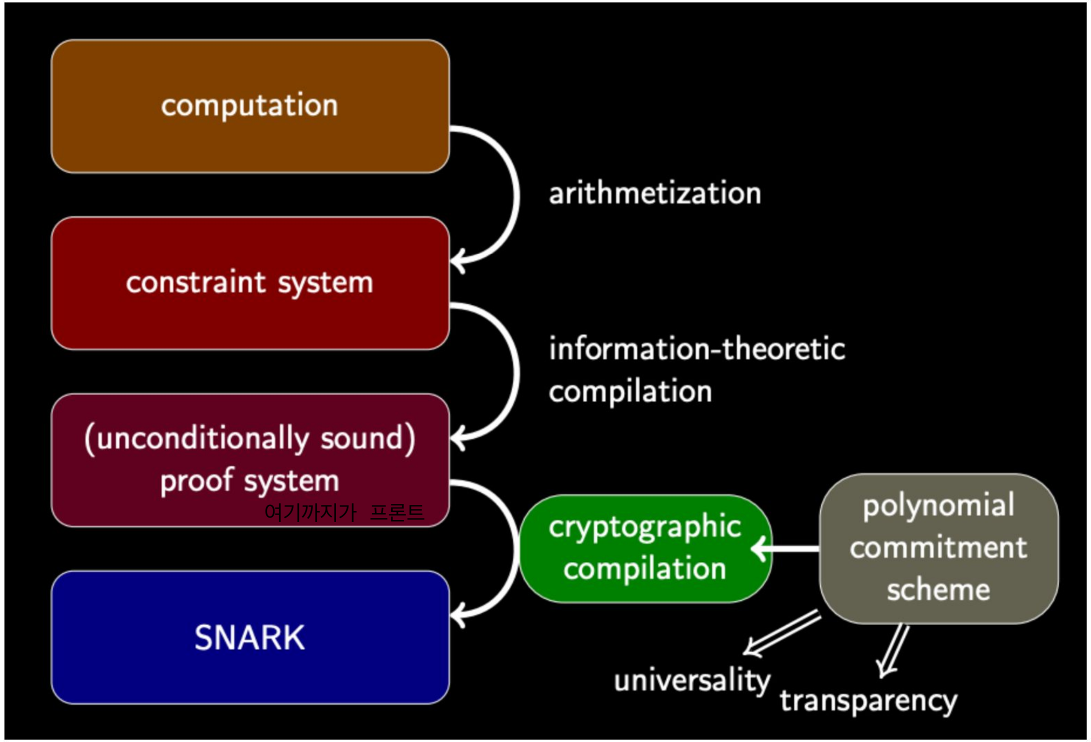
증명을 만드는데 수월



CRS (Common Reference String)

- ❖SRS
  - ❖Structured Reference String
  - ❖Trusted setup
  - ❖Universal (and Updatable)
- ❖URS
  - ❖Uniform Random String (or RRS : Random Reference String)
  - ❖Transparent

Universal / Transparent SNARK



Sonic, Marlin, Plonk vs. Kate pairing, DARK, FRI 와의 관계

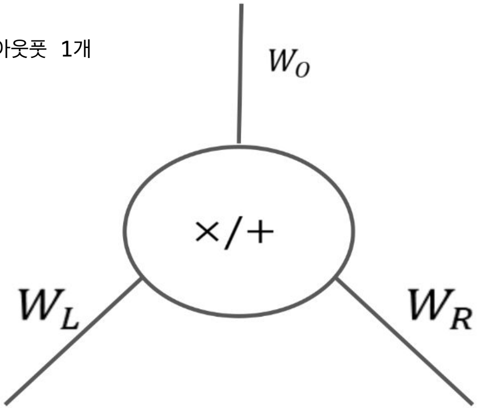
- ❖ Sonic, Marlin, Plonk
  - ❖ Circuit (함수)를 Polynomial (다항식)으로 변환시키는 scheme
  - ❖ 일종의 Front-end
- ❖Kate pairing, DARK, FRI
  - ❖ 긴 길이의 Polynomial을 짧은 값의 interaction을 통해서 증명하는 scheme
  - ❖ Polynomial commitment scheme
  - ❖ 일종의 Back-end
- ❖ 조합을 통해서 다양한 증명법 가능

예 : Plonk

곱셈 게이트와 덧셈 게이트의 표현

- 와이어 값 : W_L, W_R, W_O
- 게이트 상수 : q_M, q_L, q_R, q_O, q_C

인풋 2개 아웃풋 1개



$$q_M \cdot W_L \cdot W_R + q_L \cdot W_L + q_R \cdot W_R + q_O \cdot W_O + q_C = 0$$

Lagrange-base 형태로의 변환

와이어 값 다항식

- $W_L(X) = \sum_{i=1}^{n} W_{L,i} L_i(X)$
- $W_L(X) = \sum_{i=1}^{n} W_{L,i} L_i(X)$
- $W_L(X) = \sum_{i=1}^{n} W_{L,i} L_i(X)$

회로 다항식

- $q_M(X) = \sum_{i=1}^{n} q_{M,i} L_i(X)$
- $q_L(X) = \sum_{i=1}^{n} q_{L,i} L_i(X)$
- $q_R(X) = \sum_{i=1}^{n} q_{R,i} L_i(X)$
- $q_O(X) = \sum_{i=1}^{n} q_{O,i} L_i(X)$
- $q_C(X) = \sum_{i=1}^{n} q_{C,i} L_i(X)$

$$q_M(X)W_L(X)W_R(X) + q_L(X)W_L(X) + q_R(X)W_R(X)$$
$$+q_O(X)W_O(X) + q_C(X) = 0 \quad mod Z_H(X)$$

증명 방법

회로 관련 Polynomial :
$q_M(X)$, $q_L(X)$, $q_R(X)$, $q_{MO}(X)$, $q_C(X)$

값 관련 Polynomials :
$w_L(X)$, $w_R(X)$, $w_O(X)$

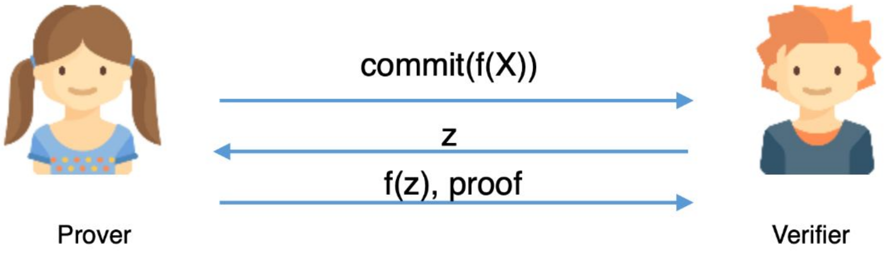
Prover

Verifier

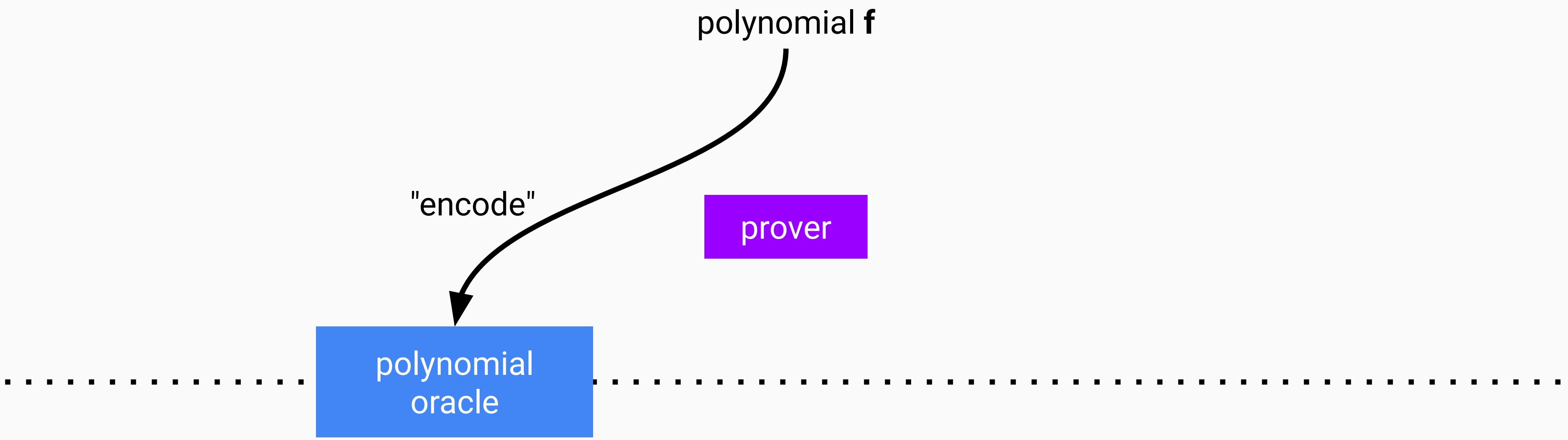
Polynomial Commitment

- ❖ Polynomial을 짧은 숫자로 표현하면서도 긴 polynomial을 나타내는 것이 맞다는 것을 보장하는 증명 방법
- ❖ Interactive Proof를 사용
- ❖ Prover가 다항식 f(X)에 대한 Polynomial commitment (Oracle)을 verifier에게 주고, Verifier는 이 Oracle에게 임의의 x (z)값에 대한 값을 질의함
- ❖ Prover는 f(z)를 계산하고, f(z)가 맞다는 proof를 함께 Verifier에게 제공
- ❖ Polynomial IOP (Interactive Oracle Proof)라고 부름

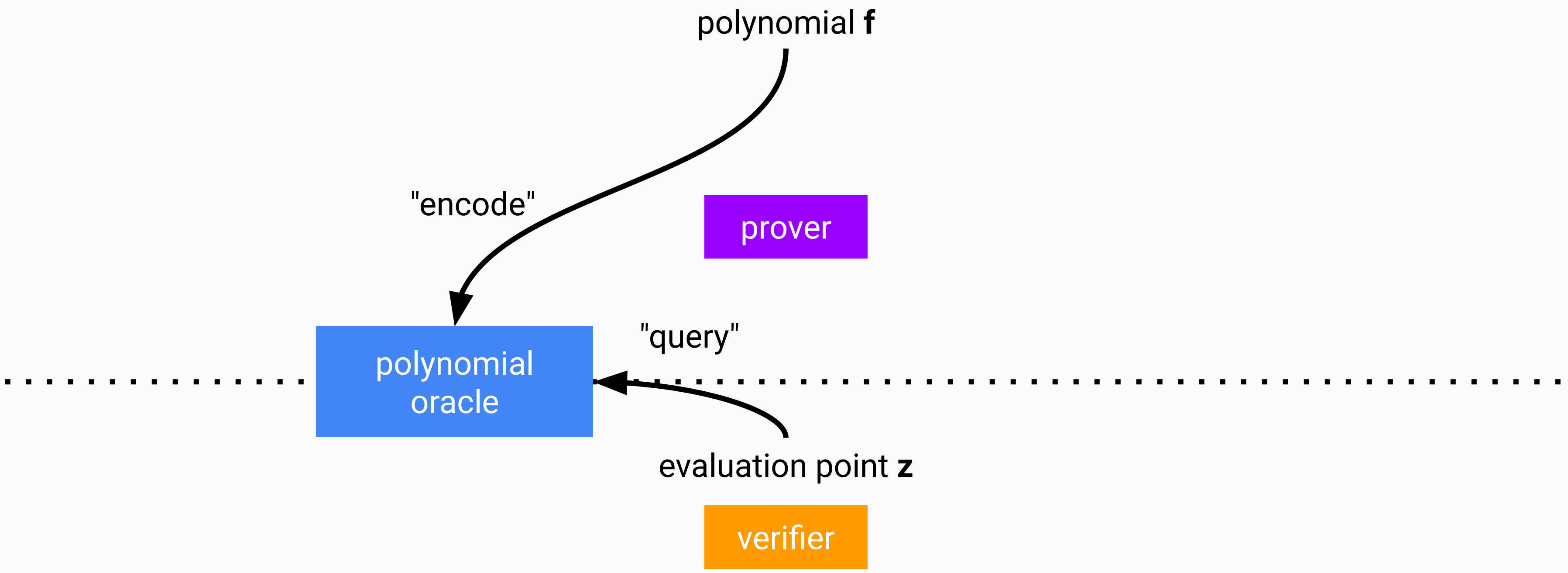
1.prover 가 먼가를 보내고
2.verifier 가 z값을보내면
3.prover가 다시 풀어서 보냄
4.verifier는 1번과 3번을 보고 관계가 있는지 확인



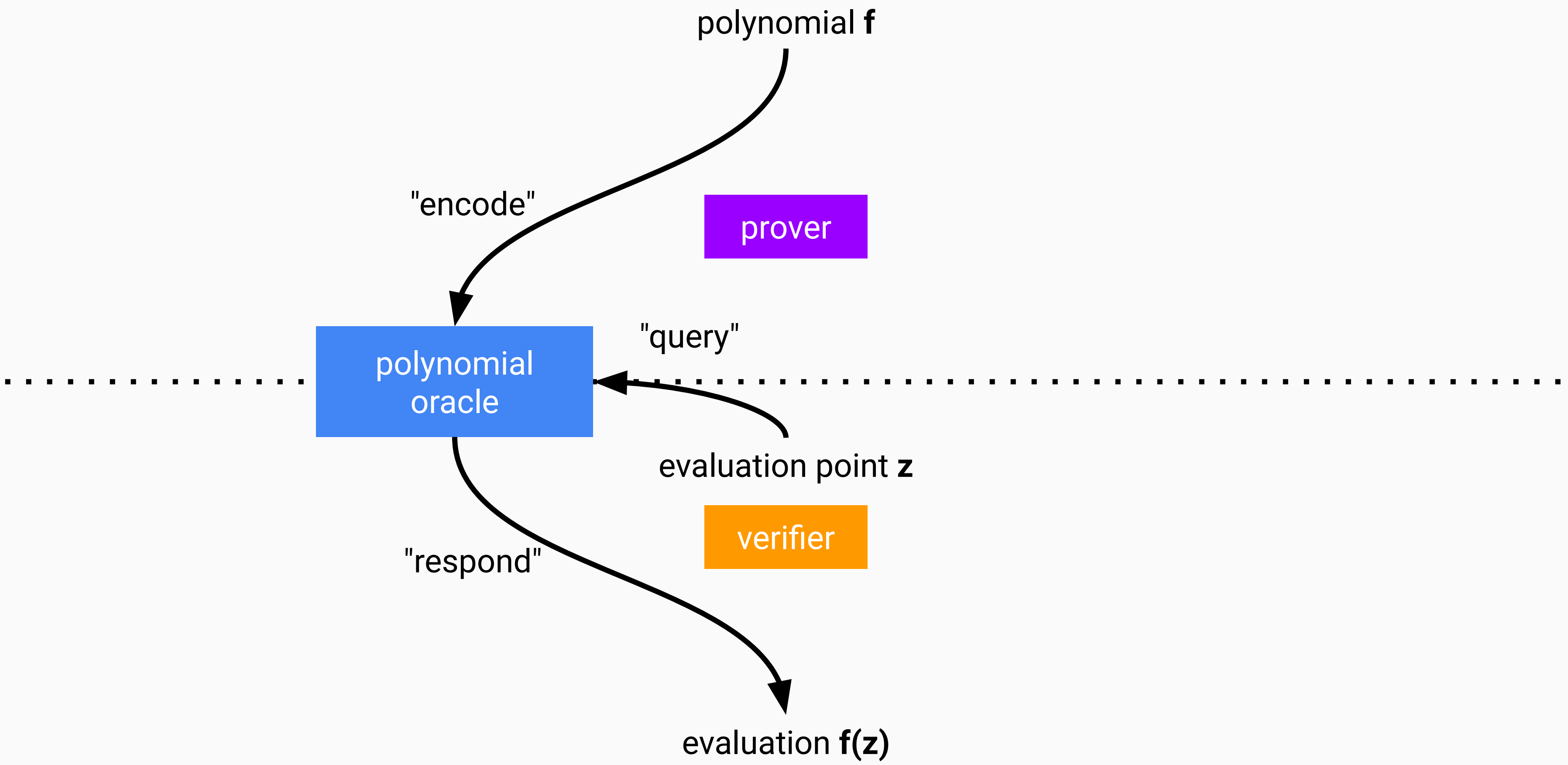
definitions



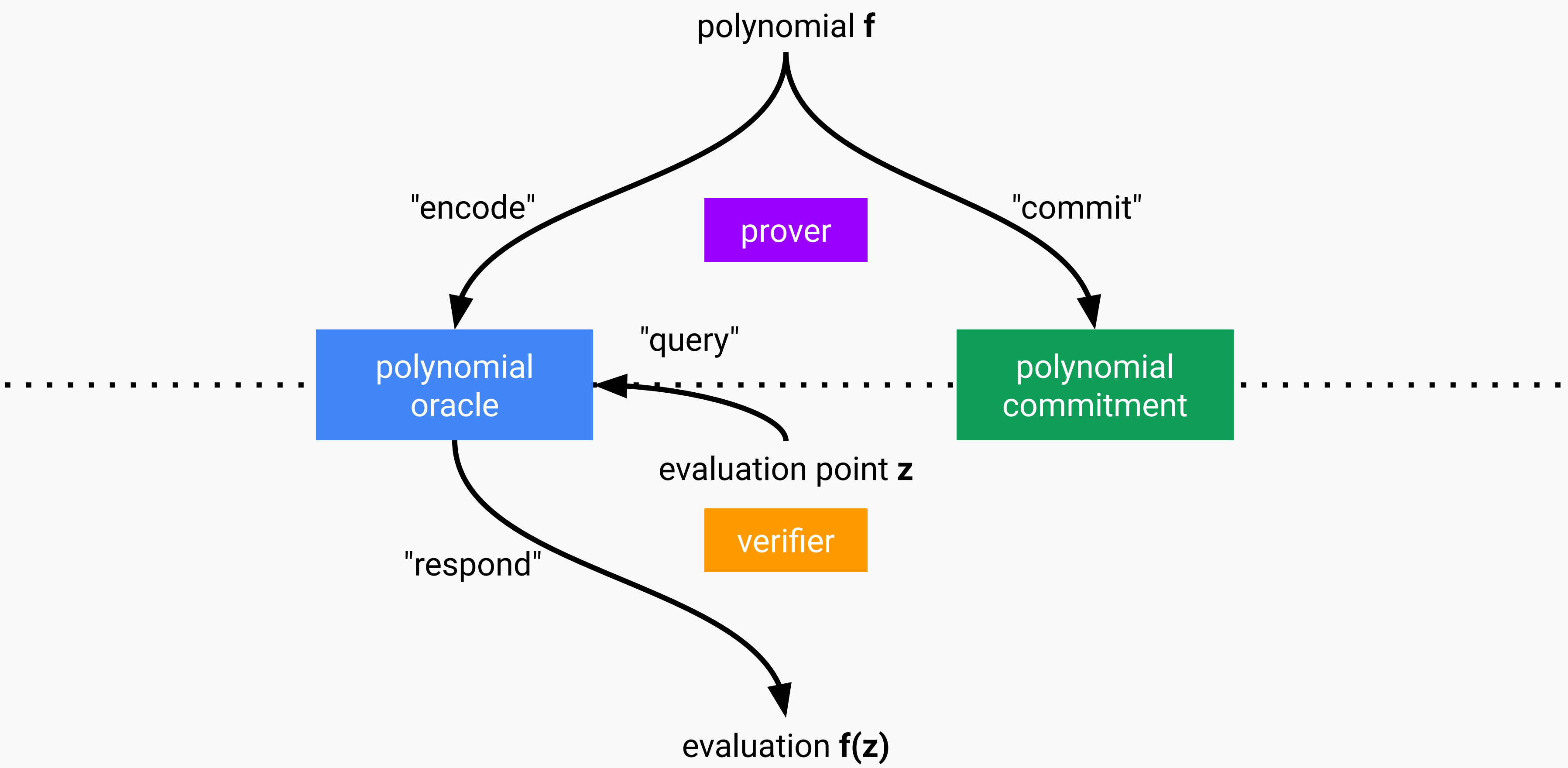
definitions



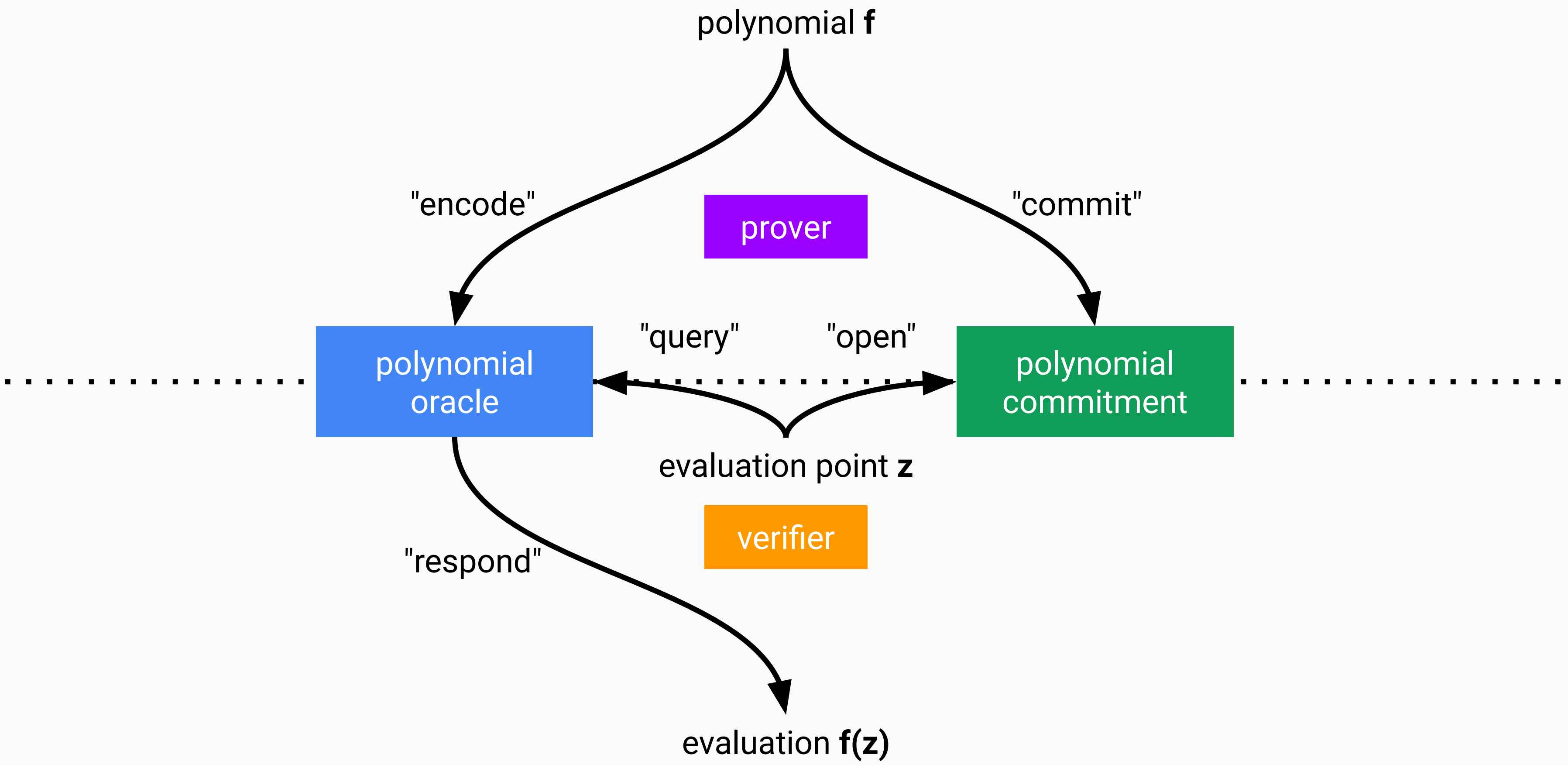
definitions



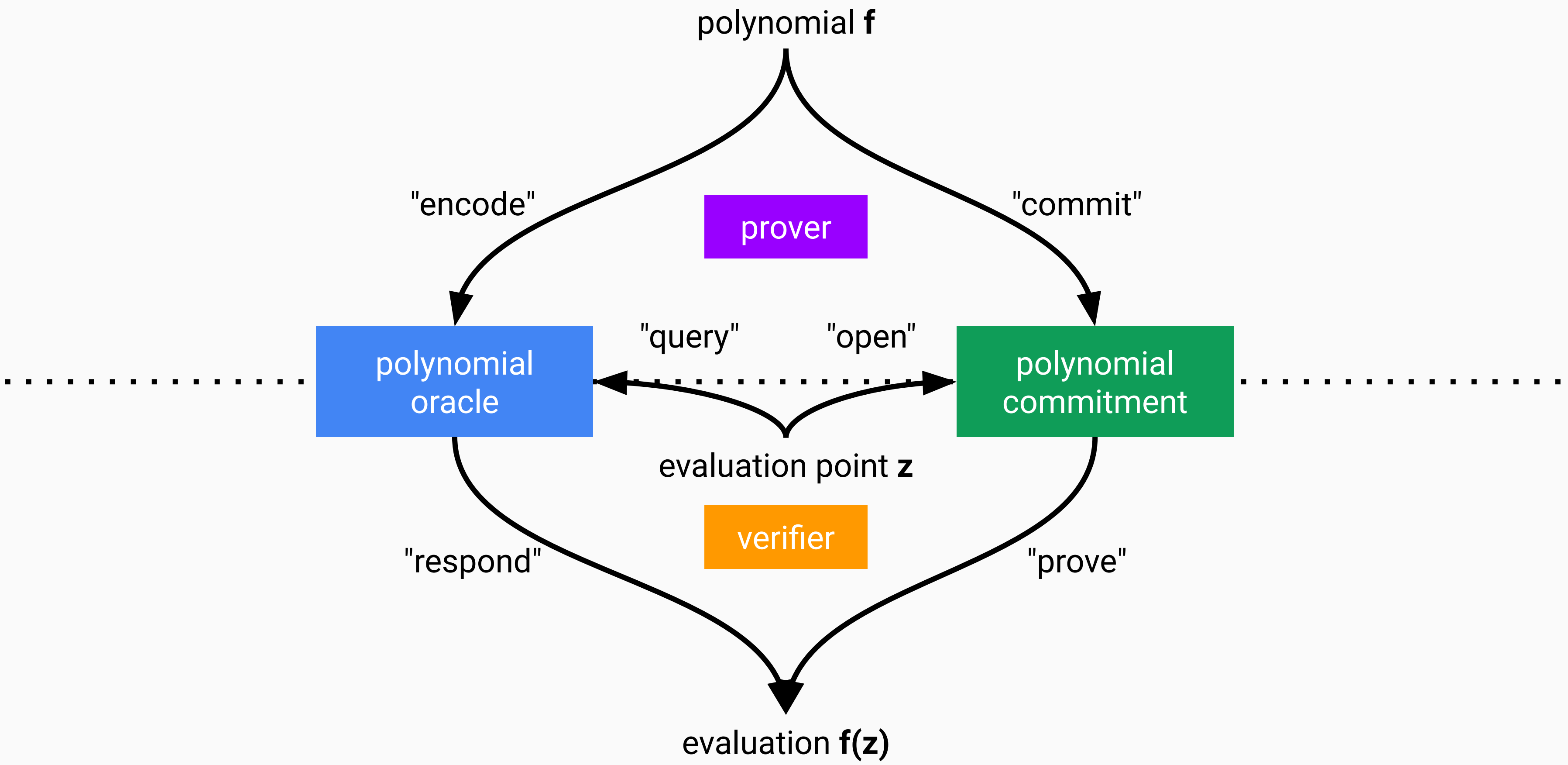
definitions



definitions



definitions



big picture

computer science

circuit

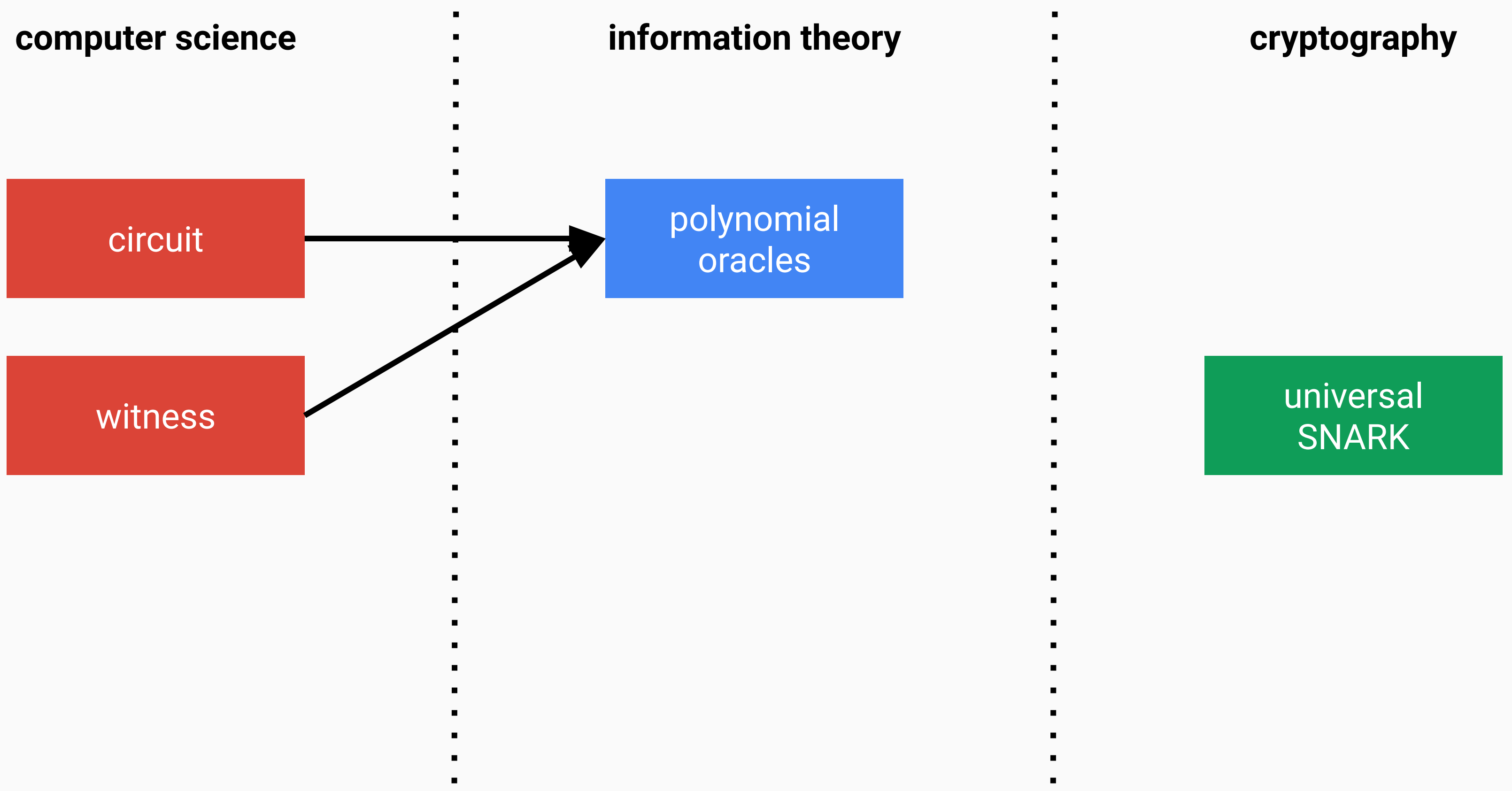
witness

information theory

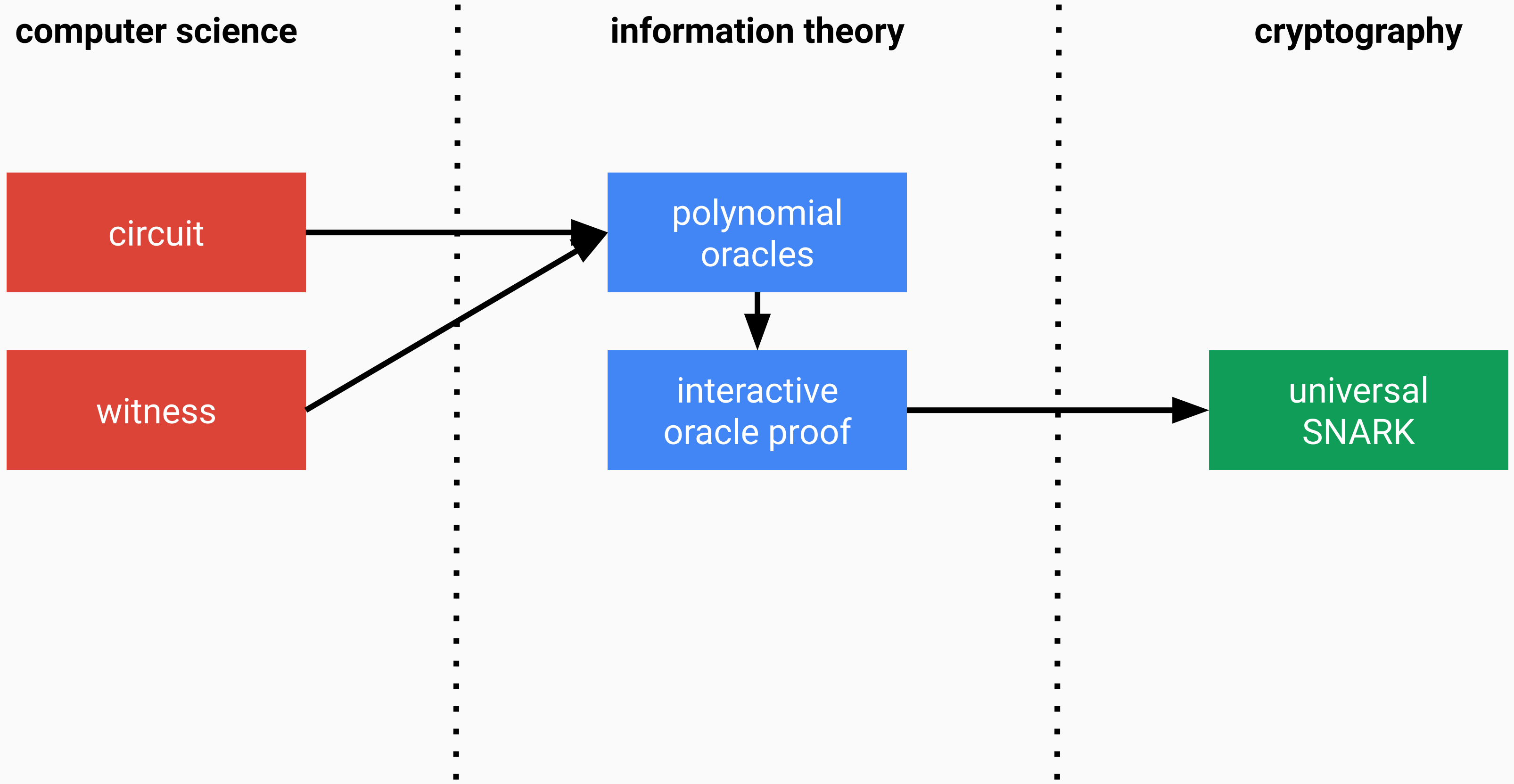
cryptography

universal SNARK

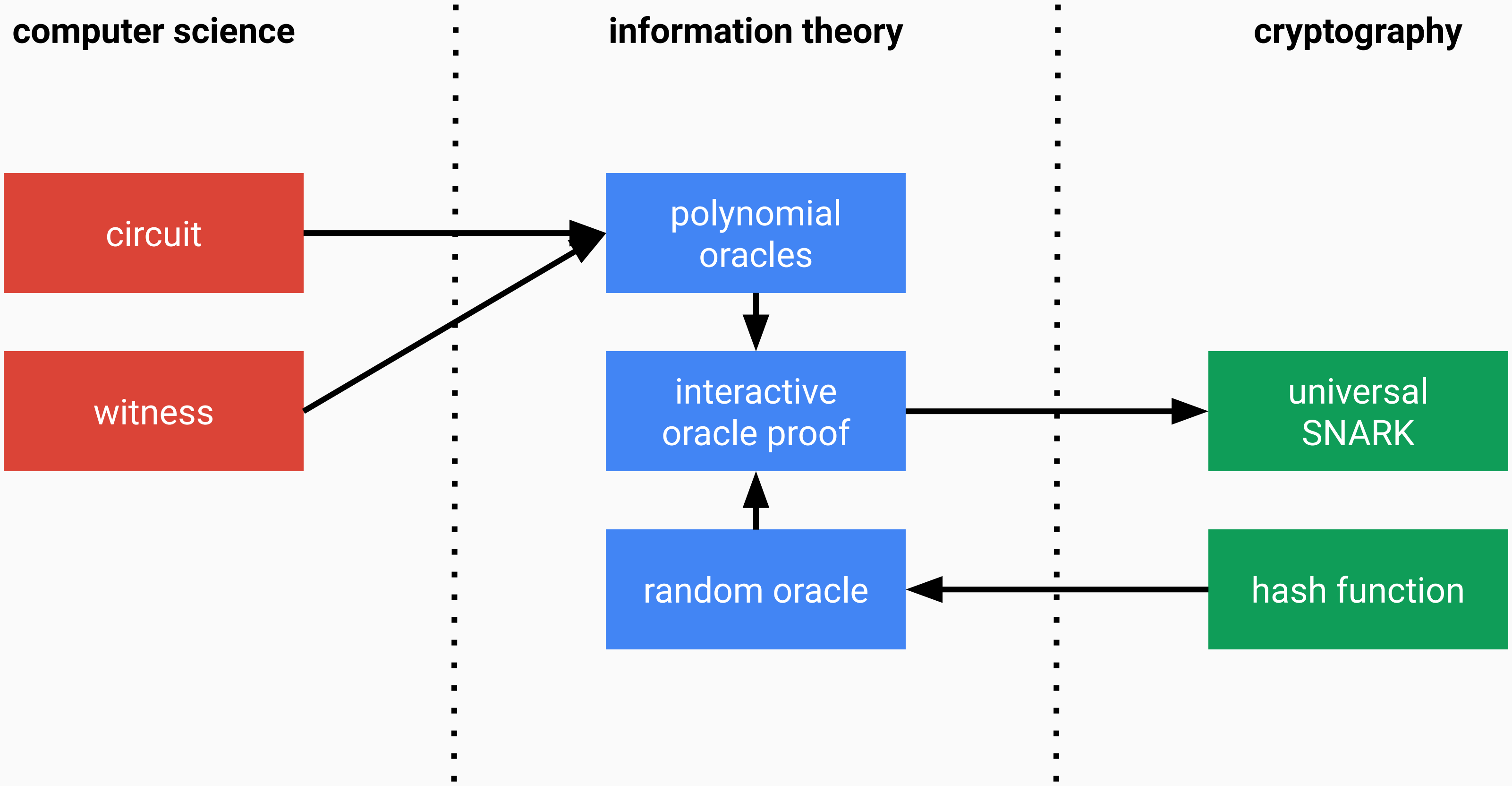
big picture


computer science
information theory
cryptography
circuit
witness
polynomial oracles
universal SNARK

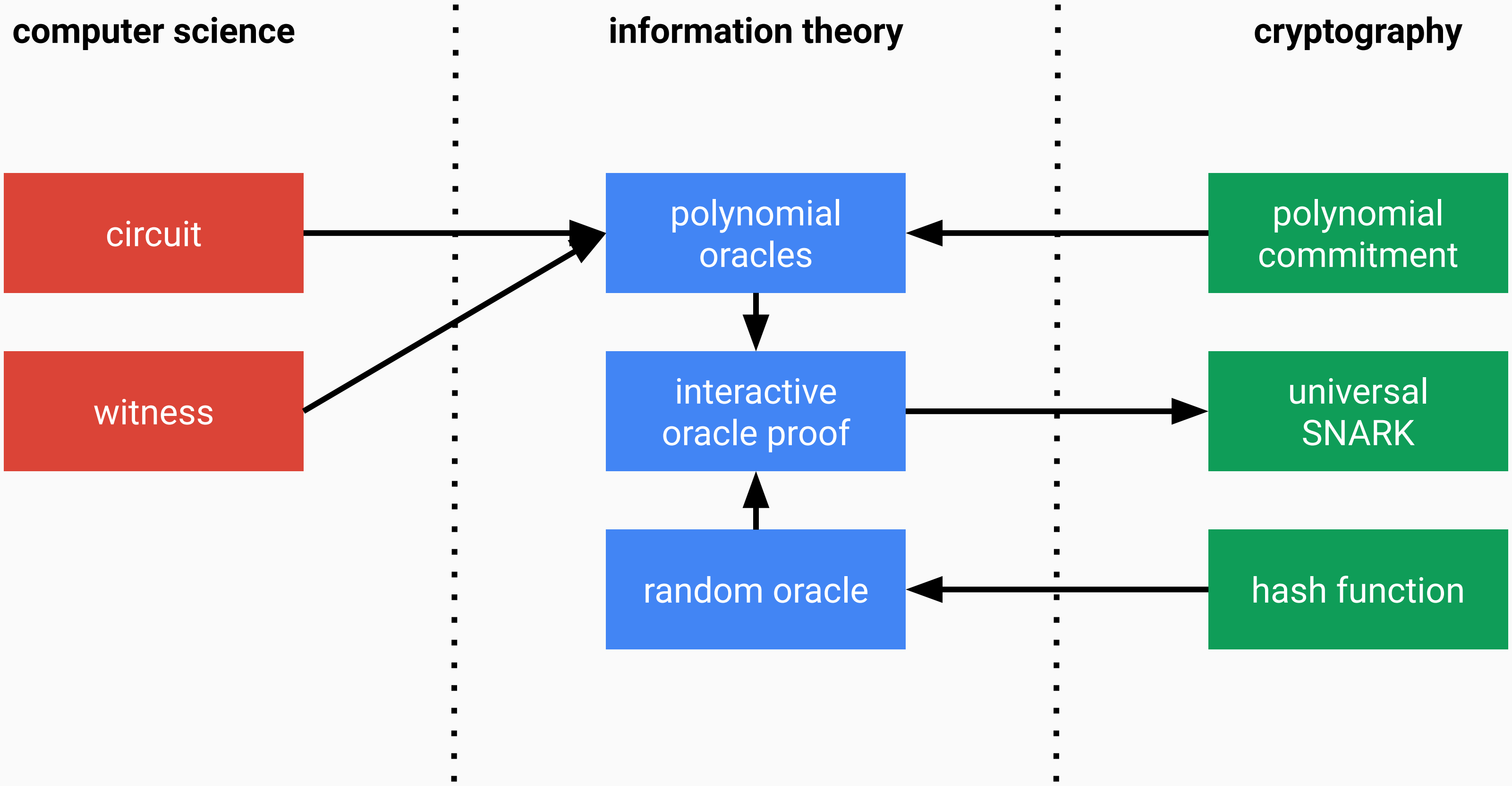
big picture


computer science
information theory
cryptography
circuit
witness
polynomial oracles
interactive oracle proof
universal SNARK

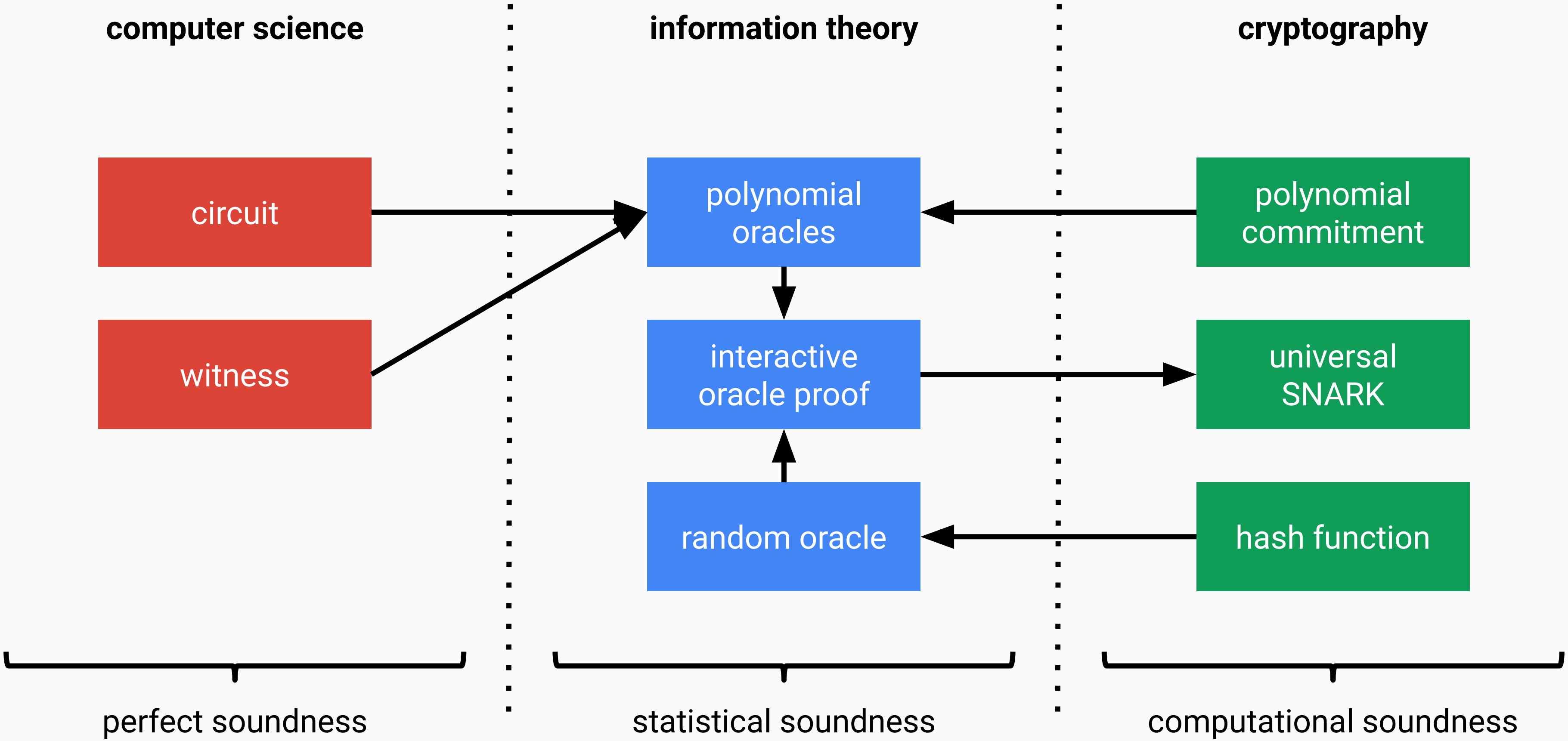
big picture


computer science
information theory
cryptography
circuit
witness
polynomial oracles
interactive oracle proof
random oracle
universal SNARK
hash function

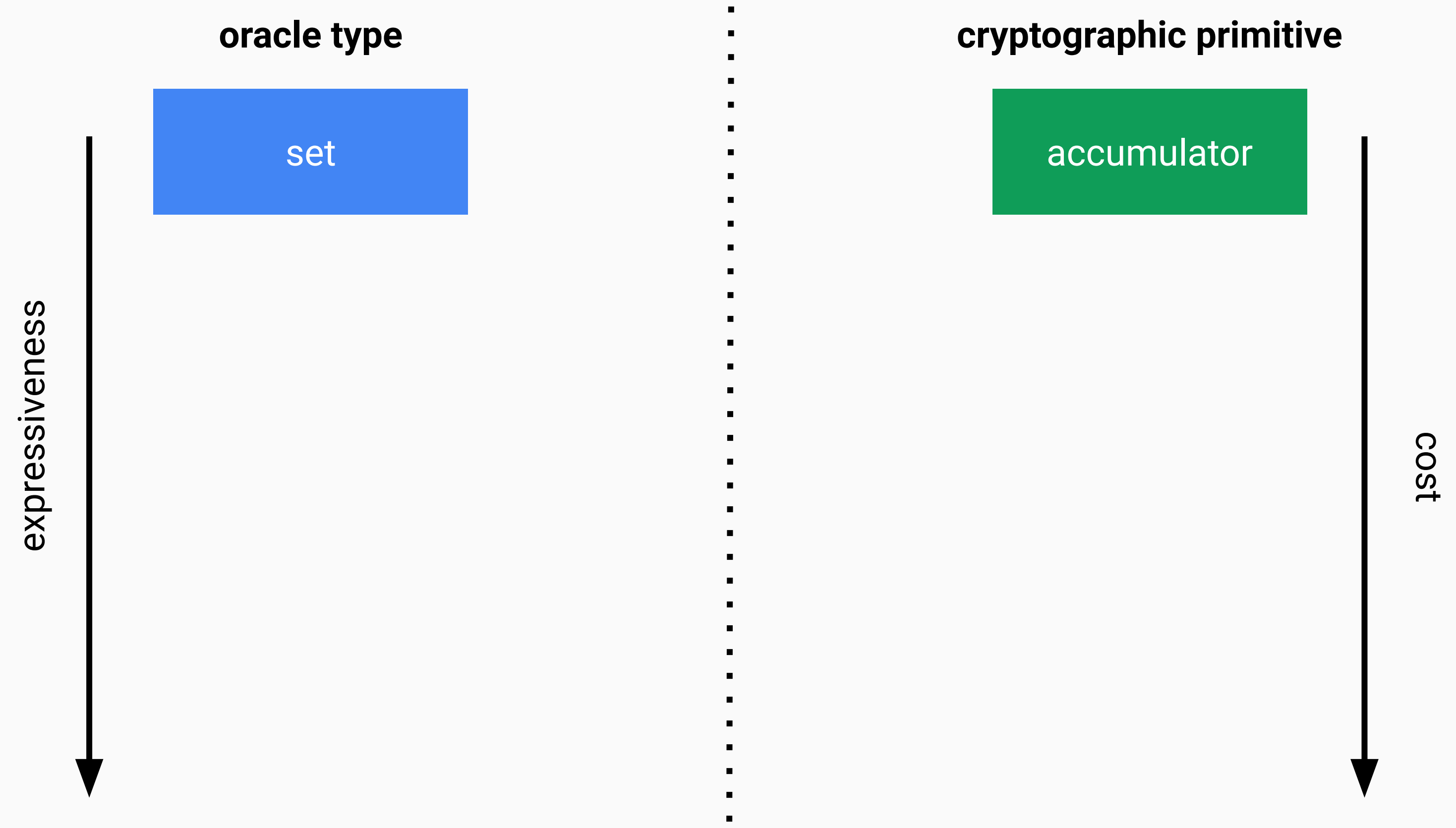
big picture


computer science
information theory
cryptography
circuit
witness
polynomial oracles
interactive oracle proof
random oracle
polynomial commitment
universal SNARK
hash function

big picture


computer science
information theory
cryptography
circuit
witness
polynomial oracles
interactive oracle proof
random oracle
polynomial commitment
universal SNARK
hash function
perfect soundness
statistical soundness
computational soundness

tug of war


oracle type
set
expressiveness
cryptographic primitive
accumulator
cost

tug of war

oracle type

- set
- vector

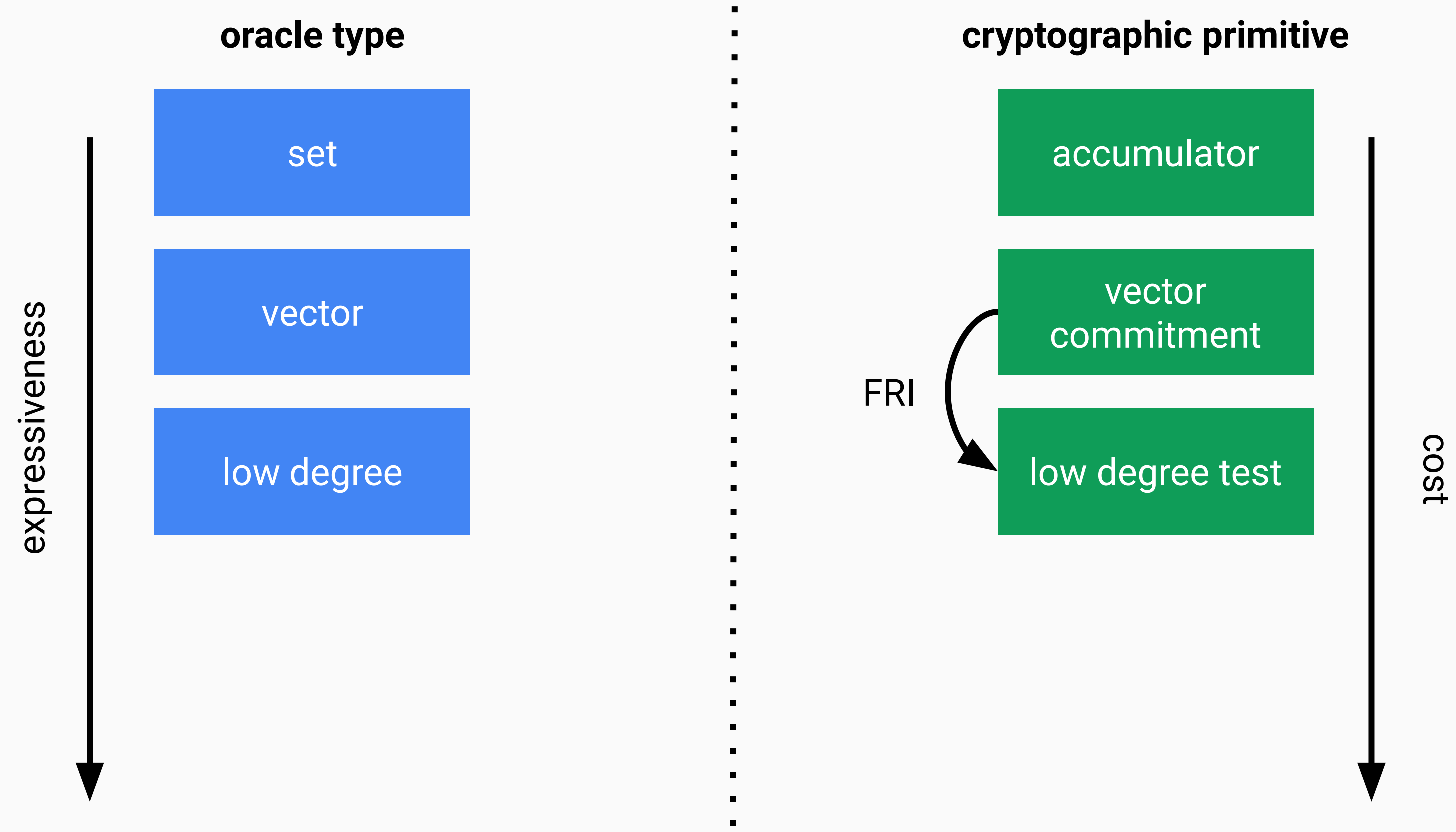
expressiveness

cryptographic primitive

- accumulator
- vector commitment

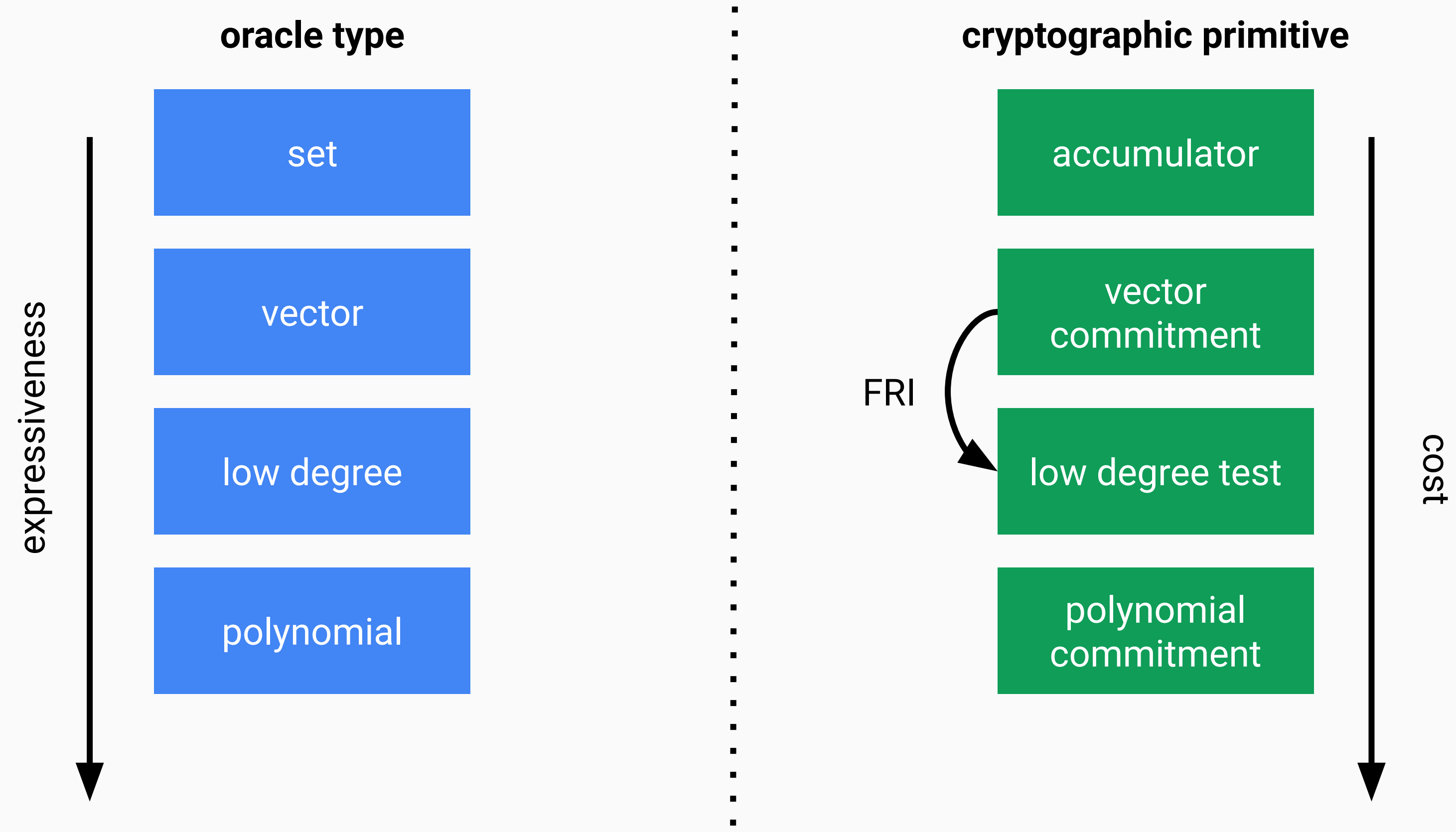
cost

tug of war

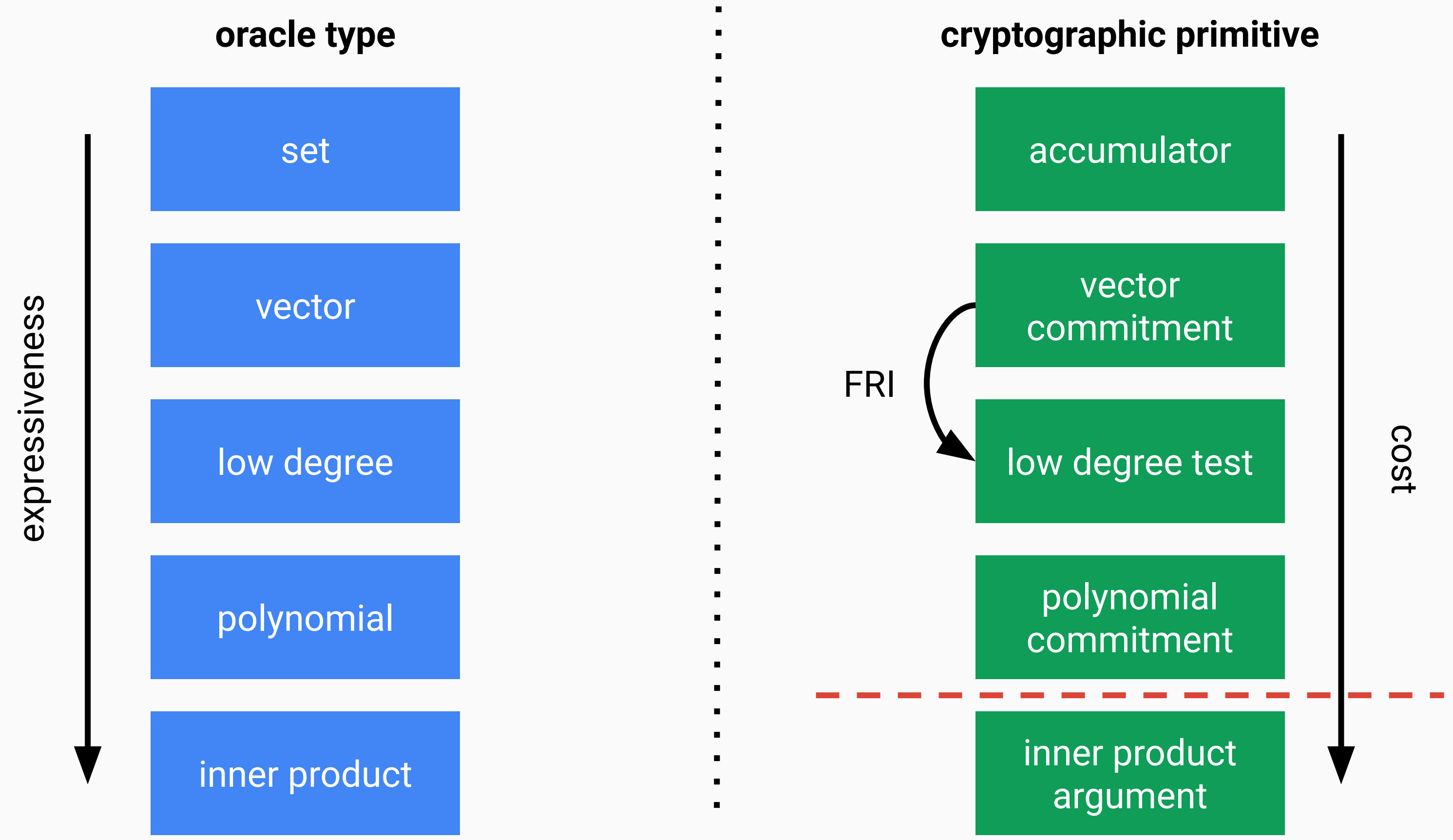


oracle type
set
vector
low degree
expressiveness
cryptographic primitive
accumulator
vector commitment
low degree test
FRI
cost

tug of war


oracle type
set
vector
low degree
polynomial
expressiveness
cryptographic primitive
accumulator
vector commitment
low degree test
polynomial commitment
FRI
cost

tug of war



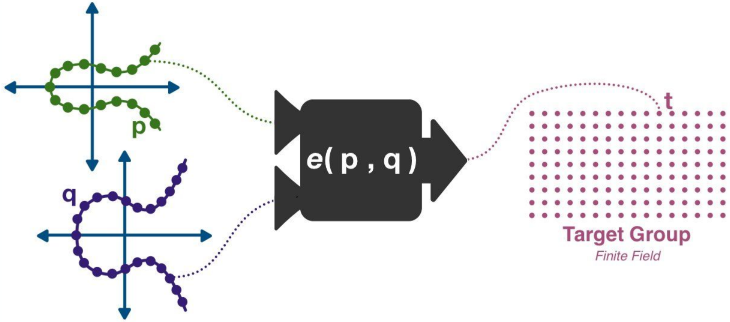
oracle type
set
vector
low degree
polynomial
inner product
expressiveness
cryptographic primitive
accumulator
vector commitment
FRI
low degree test
polynomial commitment
inner product argument
cost

Elliptic Curve Pairings

Elliptic curve pairings are the elliptic point equivalent of multiplication

$$e(p, q) = t \approx p * q = t$$

An elliptic curve pairing is a function that takes a pair of elliptic curve points returns an element of some other group, called the target group



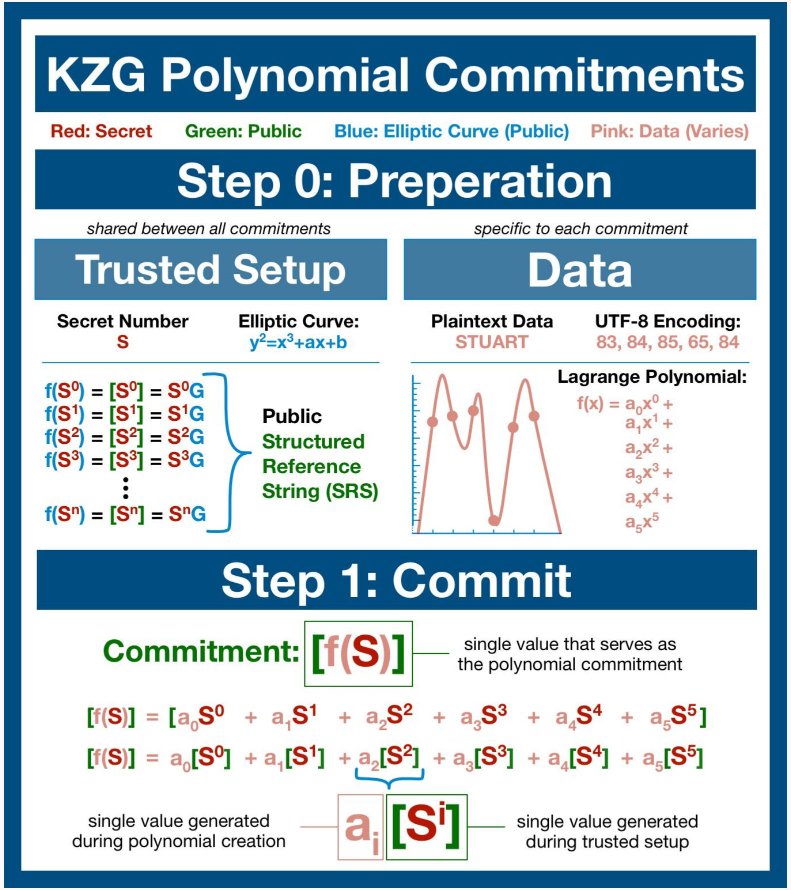
Think of a pairing as a black box that takes elliptic points. Pairings cannot be uses consecutively; the target group points don't match the input points

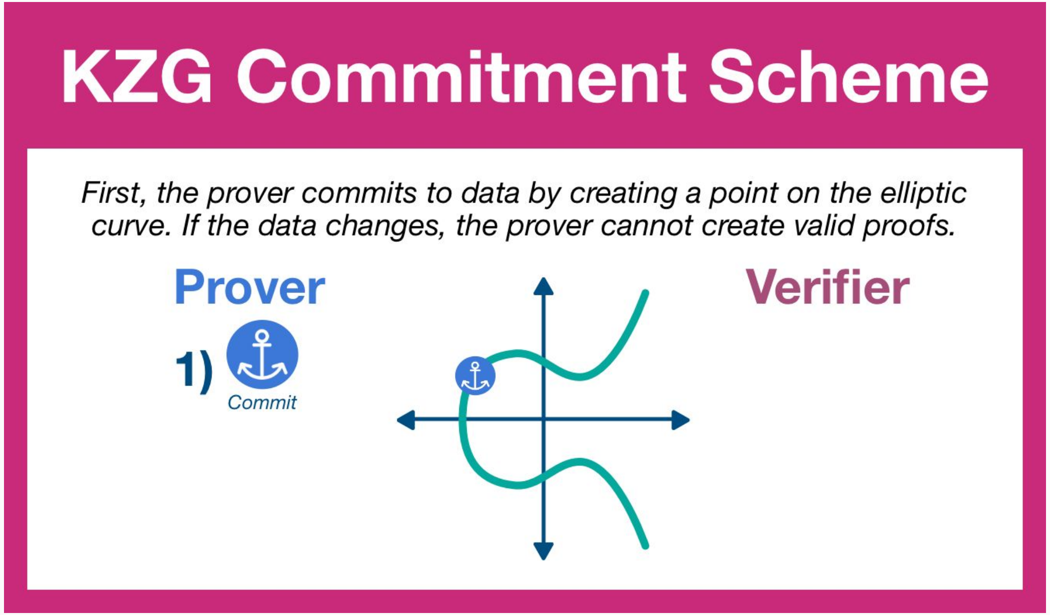
Elliptic curve pairings are bilinear, holding to the following property:

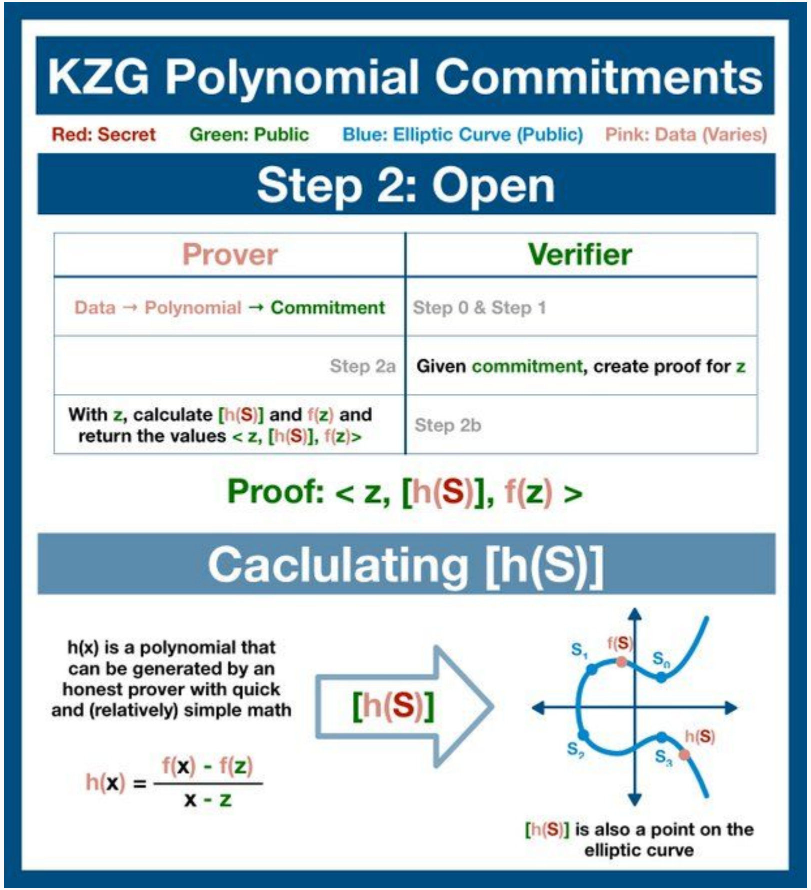
$$e(p+r, q) = e(p, q) * e(r, q)$$

$$e(p, q+r) = e(p, q) * e(p, r)$$

Translation: you can pull additive component out of a pairing by multiplication

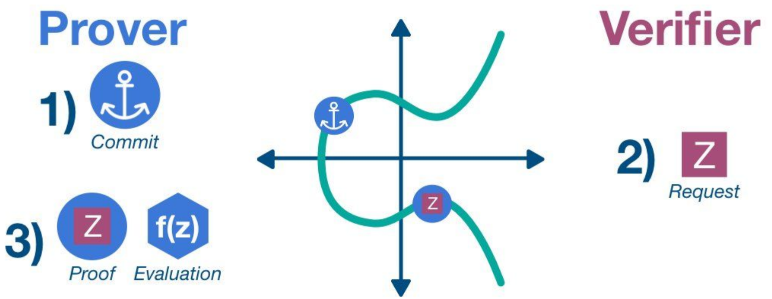

KZG Polynomial Commitments
Red: Secret
Green: Public
Blue: Elliptic Curve (Public)
Pink: Data (Varies)
Step 0: Preperation
shared between all commitments
specific to each commitment
Trusted Setup
Data
Secret Number
S
Elliptic Curve:
$y^2=x^3+ax+b$
Plaintext Data
STUART
UTF-8 Encoding:
83, 84, 85, 65, 84
$f(S^0) = [S^0] = S^0G$
$f(S^1) = [S^1] = S^1G$
$f(S^2) = [S^2] = S^2G$
$f(S^3) = [S^3] = S^3G$
$f(S^n) = [S^n] = S^nG$
Public Structured Reference String (SRS)
Lagrange Polynomial:
$f(x) = a_0x^0 + a_1x^1 + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5$
Step 1: Commit
Commitment: [f(S)]
single value that serves as the polynomial commitment
$[f(S)] = [a_0S^0 + a_1S^1 + a_2S^2 + a_3S^3 + a_4S^4 + a_5S^5]$
$[f(S)] = a_0[S^0] + a_1[S^1] + a_2[S^2] + a_3[S^3] + a_4[S^4] + a_5[S^5]$
single value generated during polynomial creation
a_i $[S^i]$
single value generated during trusted setup


KZG Commitment Scheme
First, the prover commits to data by creating a point on the elliptic curve. If the data changes, the prover cannot create valid proofs.
Prover
Verifier
1)
Commit

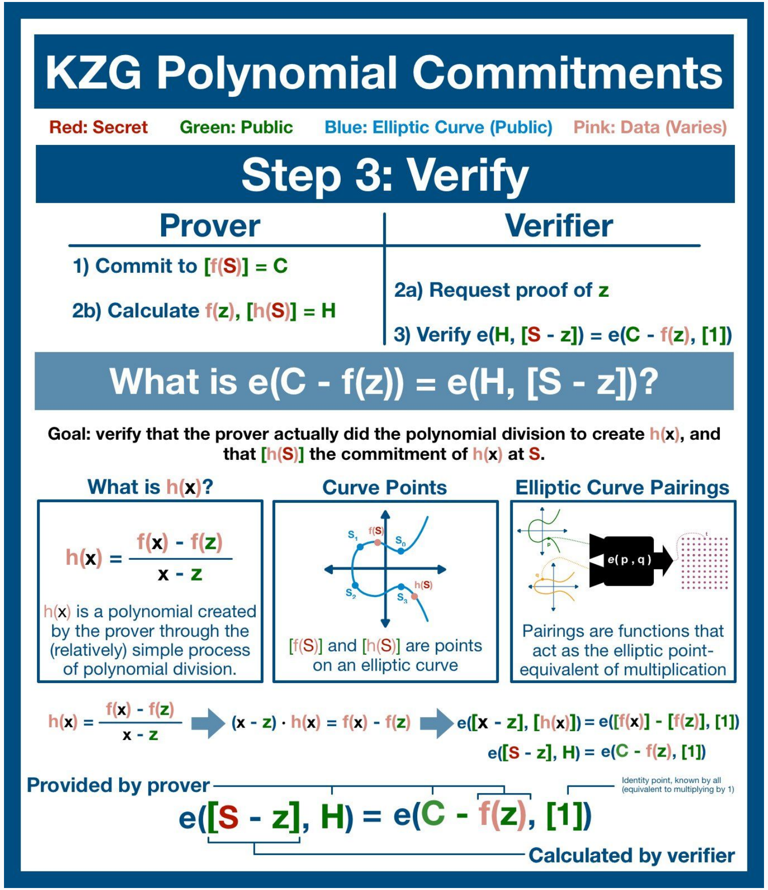


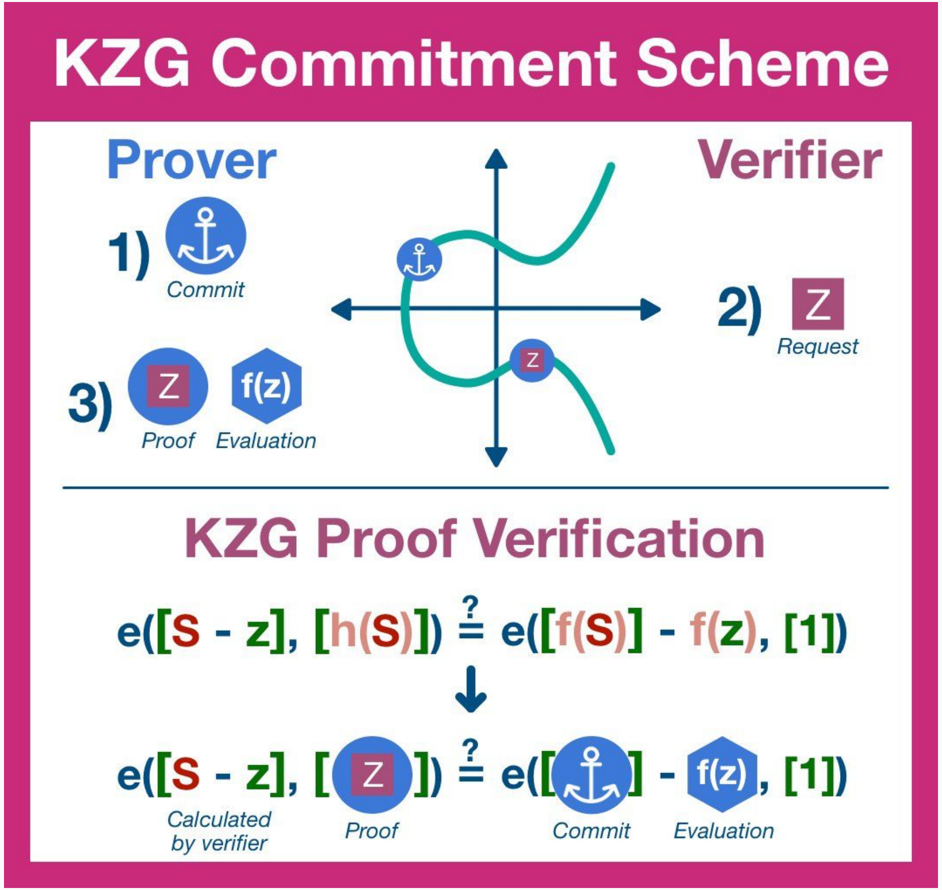
KZG Commitment Scheme

First, the prover commits to data by creating a point on the elliptic curve. If the data changes, the prover cannot create valid proofs.



Next, the verifier gives a data point. The prover builds a new elliptic curve point and a polynomial evaluation around that point.


KZG Polynomial Commitments
Red: Secret
Green: Public
Blue: Elliptic Curve (Public)
Pink: Data (Varies)
Step 3: Verify
Prover
Verifier
1) Commit to [f(S)] = C
2a) Request proof of z
2b) Calculate f(z), [h(S)] = H
3) Verify e(H, [S - z]) = e(C - f(z), [1])
What is e(C - f(z)) = e(H, [S - z])?
Goal: verify that the prover actually did the polynomial division to create h(x), and that [h(S)] the commitment of h(x) at S.
What is h(x)?
h(x) = (f(x) - f(z)) / (x - z)
h(x) is a polynomial created by the prover through the (relatively) simple process of polynomial division.
Curve Points
[f(S)] and [h(S)] are points on an elliptic curve
Elliptic Curve Pairings
e(p, q)
Pairings are functions that act as the elliptic point-equivalent of multiplication
h(x) = (f(x) - f(z)) / (x - z)
(x - z) · h(x) = f(x) - f(z)
e([x - z], [h(x)]) = e([f(x)] - [f(z)], [1])
e([S - z], H) = e(C - f(z), [1])
Provided by prover
Identity point, known by all (equivalent to multiplying by 1)
e([S - z], H) = e(C - f(z), [1])
Calculated by verifier


KZG Commitment Scheme
Prover
Verifier
1) Commit
2) Z Request
3) Z Proof
f(z) Evaluation
KZG Proof Verification
e([S - z], [h(S)]) ≟ e([f(S)] - f(z), [1])
e([S - z], [Z]) ≟ e([Commit] - f(z), [1])
Calculated by verifier
Proof
Commit
Evaluation

part 1—context

part 2—landscape

part 3—mechanics

part 4—gadgets

setup and commitment

	FRI	KZG	DARK	Bulletproof
	hash function	pairing group	unknown order group	discrete log group
setup	**H** hash function **w** in **F** root of unity	G_1, G_2 groups g_1, g_2 generators **e** pairing **s** in **F** secret	**G** unknown order **g** in **G** random **q** large integer	**G** elliptic curve g_i in **G** independent
commitment	$root(f(w^0), ..., f(w^{kd}))$	$a_0 s^0 g_1 + ... + a_n s^d g_1$	$a_0 q^0 g + ... + a_d q^d g$	$a_0 g_0 + ... + a_d g_d$

algebraic
(with linear homomorphism)

comparing setups

	FRI	KZG	DARK			Bulletproof
	hash function	**pairing group**	**RSA group**	**class group**	**Jacobian group**	**discrete log group**
transparent						
succinct						
unbounded						
updatable						
post-quantum						

asymptotic performance

max(commitment size, opening proof size)

	FRI	KZG	DARK	Bulletproof
	hash function	pairing group	unknown order group	discrete log group
size	$O(\log^2(d))$	$O(1)$	$O(\log(d))$	$O(\log(d))$
verifier time	$O(\log^2(d))$	$O(1)$	$O(\log(d))$	$O(d)$
prover time	$O(d*\log(d))$	$O(d)$	$O(d)$	$O(d)$

max(commitment time, opening time)

commitment size (with security parameter **λ** and **d** ≪ **λ**)

subexponential attacks
(sieve based)

polynomial attack

FRI	KZG	DARK			Bulletproof
hash function	pairing group	RSA group	class group	Jacobian group	discrete log group
$O(1) * O(\lambda)$	$O(1) * O(\lambda^3)$	$O(1) * O(\lambda^3)$	$O(1) * O(\lambda^2)$	???	$O(1) * O(\lambda)$

commitment size (with security parameter **λ** and **d** ≪ **λ**)

subexponential attacks
(sieve based)

polynomial attack

FRI	KZG	DARK			Bulletproof
hash function	**pairing group**	**RSA group**	**class group**	**Jacobian group**	**discrete log group**
$O(1) * O(\lambda)$	$O(1) * O(\lambda^3)$	$O(1) * O(\lambda^3)$	$O(1) * O(\lambda^2)$	???	$O(1) * O(\lambda)$

target
128 bits
security

SHA256	**BLS12-381**	**RSA**	**class group**	**Jacobian group**	**secp256k1**
256 bits	381 bits	4,096 bits	1,827 bits	303 bits	256 bits
(32 bytes)	(48 bytes)	(512 bytes)	(229 bytes)	(38 bytes)	(32 bytes)

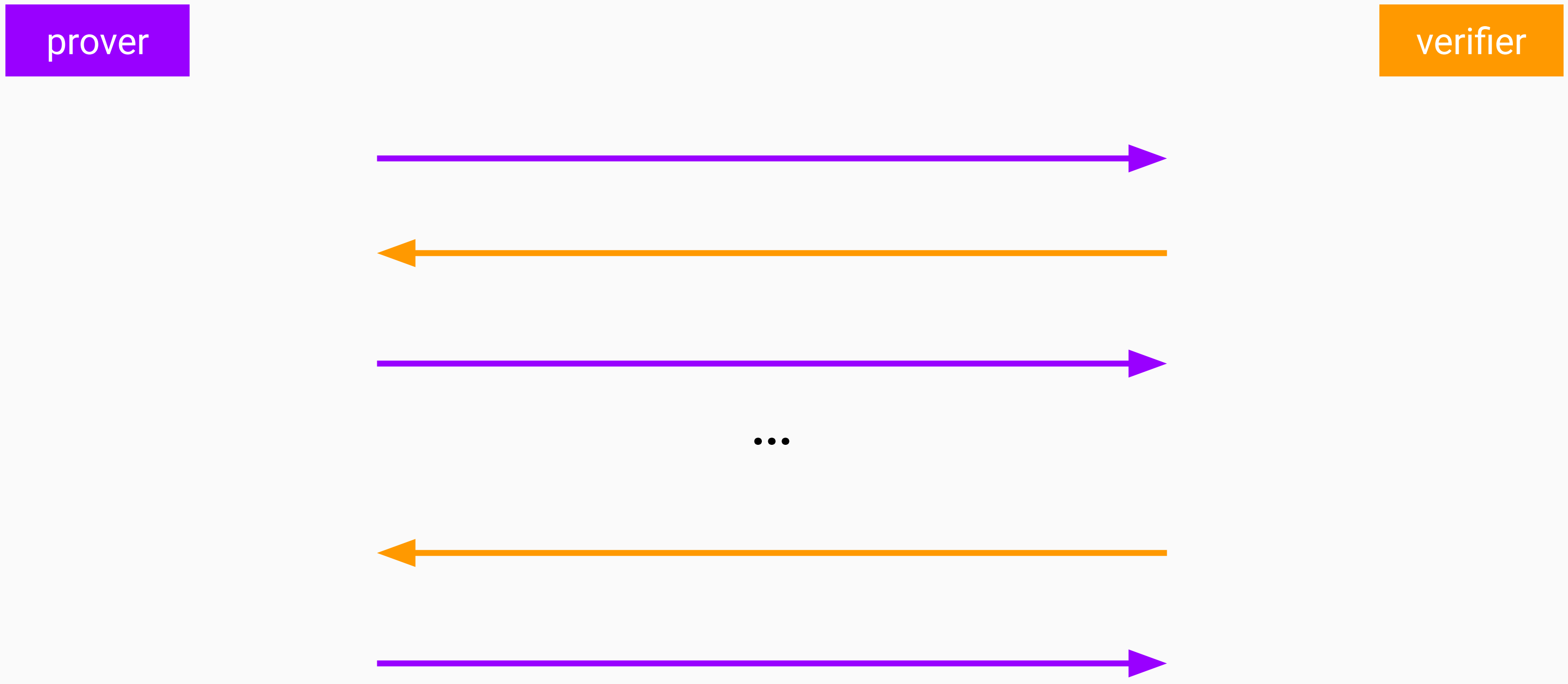
part 1—context

part 2—landscape

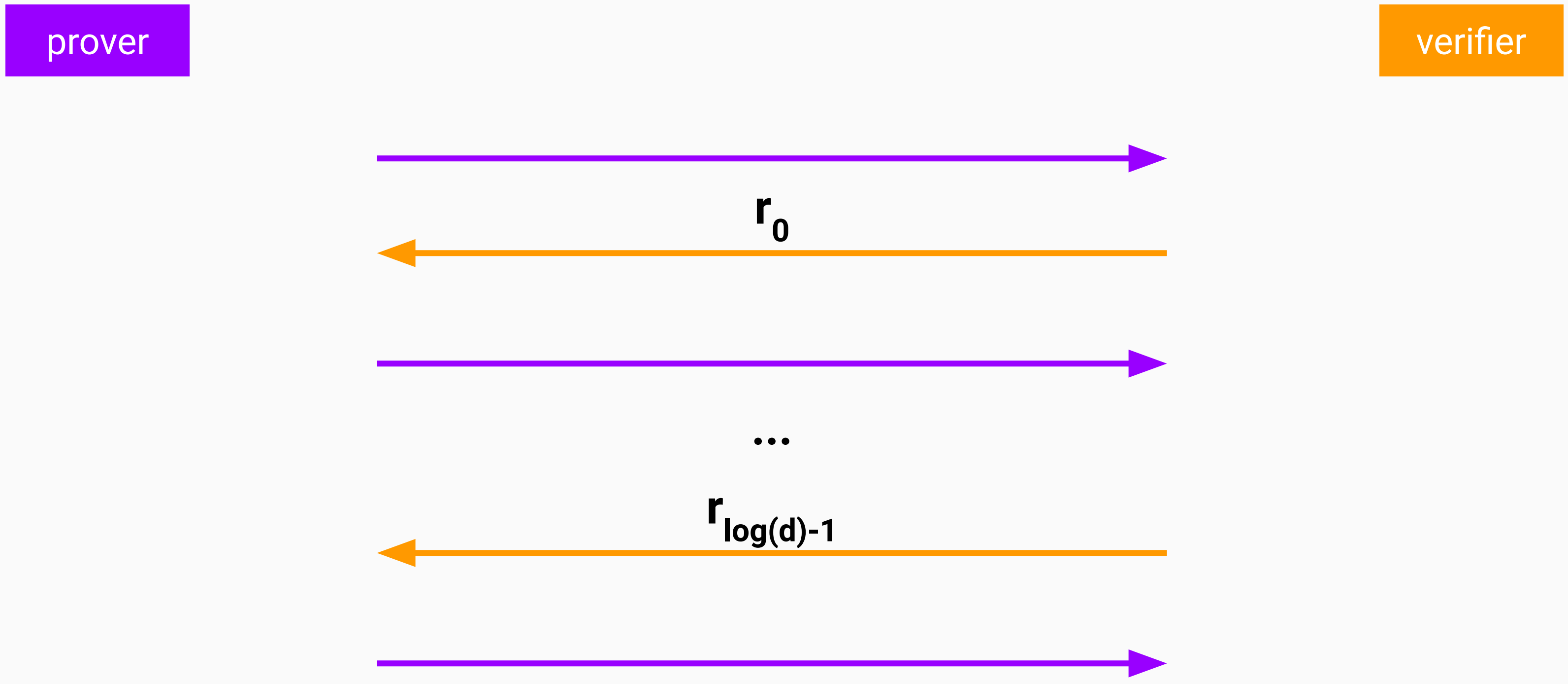
part 3—mechanics

part 4—gadgets

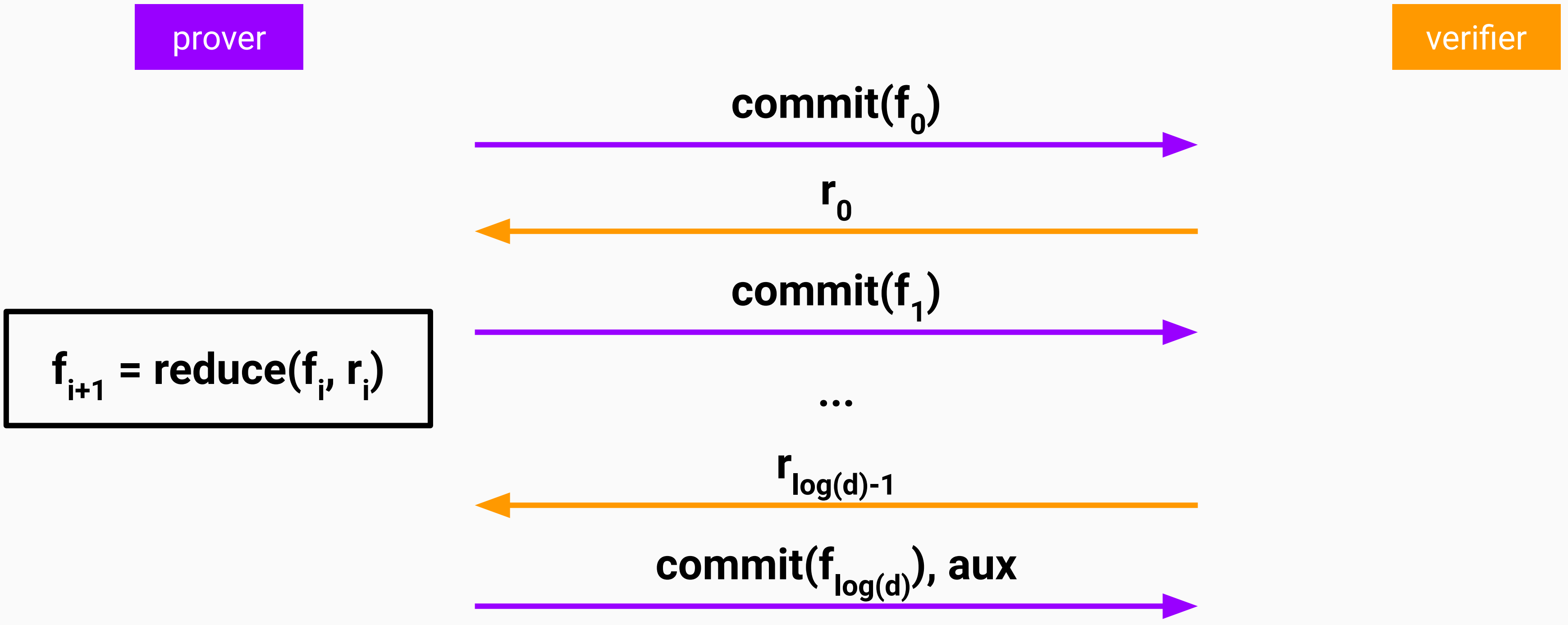
commit-reduce low degree test



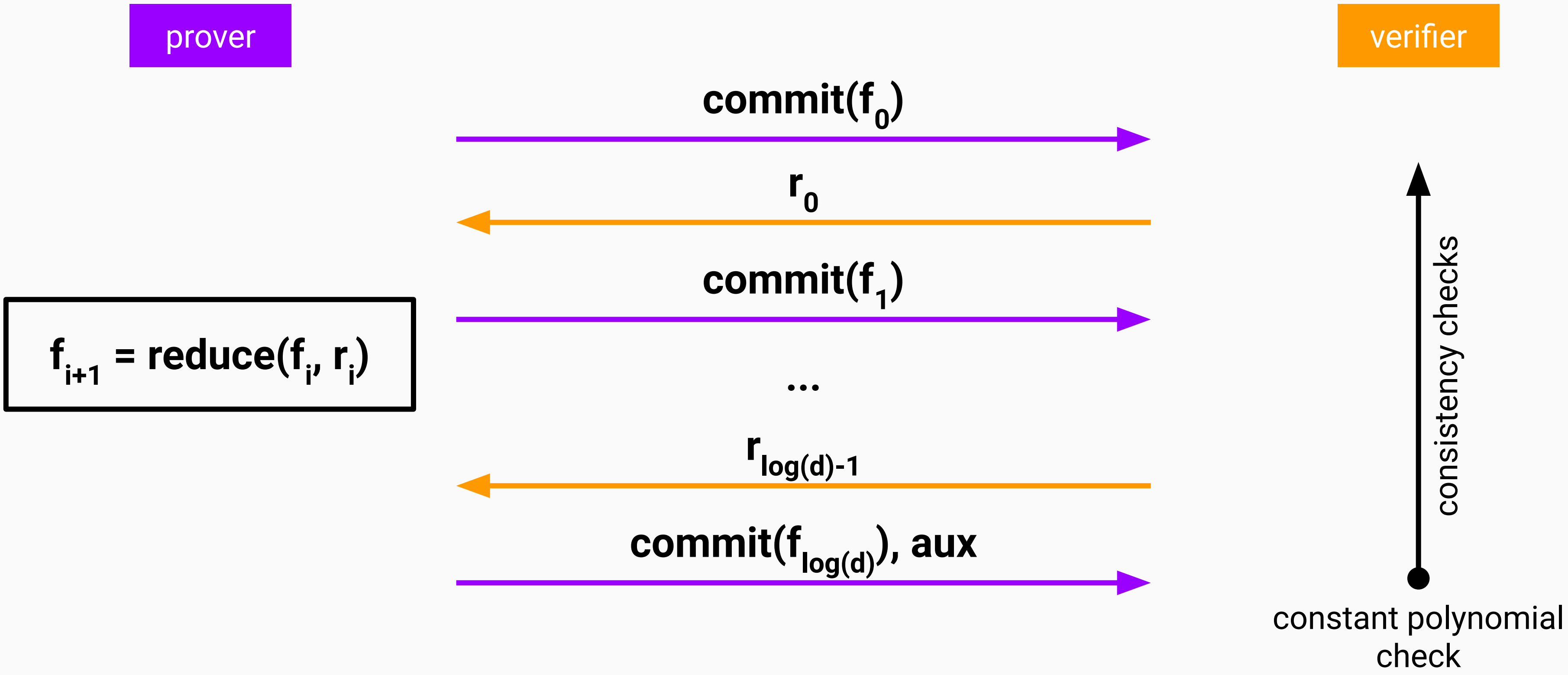
commit-reduce low degree test



commit-reduce low degree test



commit-reduce low degree test



even-odd and left-right decomposition

$$f(X) = even(f)(X^2) + X*odd(f)(X^2)$$

even-odd decomposition

$$f(X) = left(f)(X) + X^{d/2}*right(f)(X)$$

left-right decomposition

decompose-reduce

$$f(X) = \text{even}(f)(X^2) + X * \text{odd}(f)(X^2)$$

even-odd decomposition

$$f(X) = \text{left}(f)(X) + X^{d/2} * \text{right}(f)(X)$$

left-right decomposition

	hash function (FRI)	UO group (DARK)	discrete log group (Bulletproof)
coefficients	even(f) + r*odd(f)	even(f) + r*odd(f)	r*left(f) + r^{-1}*right(f)
basis	N/A	g	r^{-1}*left(g) + r*right(g)

consistency checks

FRI (hash function)

$$2zf_{i+1}(z^2) \stackrel{?}{=} z(f_i(z) + f_i(-z)) + r_i(f_i(z) - f_i(-z))$$

DARK (UO group)

Bulletproof (discrete log)

consistency checks

FRI (hash function)

$$2z f_{i+1}(z^2) \stackrel{?}{=} z(f_i(z) + f_i(-z)) + r_i(f_i(z) - f_i(-z))$$

DARK (UO group)

$$\mathrm{commit}(f_{i+1}) \stackrel{?}{=} \mathrm{commit}(\mathrm{even}(f_i)) + r_i * q * \mathrm{commit}(\mathrm{odd}(f_i))$$

Bulletproof (discrete log)

consistency checks

FRI (hash function)	DARK (UO group)	Bulletproof (discrete log)
$2zf_{i+1}(z^2)$	$\text{commit}(f_{i+1})$	$\text{commit}(f_{i+1})$
?=	?=	?=
$z(f_i(z) + f_i(-z))$	$\text{commit}(\text{even}(f_i))$	$\text{commit}(f_i)$
+	+	+
$r_i(f_i(z) - f_i(-z))$	$r_i * q * \text{commit}(\text{odd}(f_i))$	$(r_i)^2 L + (r_i)^{-2} R$

quotient argument openings

$$\boxed{f(X) - f(z) = q(X)(X - z)}$$

FRI (hash function)

KZG10 (pairing group)

$(f(X) - f(z))/(X - z)$ low degree proof

(within unique decoding radius)

quotient argument openings

$$\boxed{f(X) - f(z) = q(X)(X - z)}$$

FRI (hash function)

$(f(X) - f(z))/(X - z)$ low degree proof

(within unique decoding radius)

KZG10 (pairing group)

$$e(\mathrm{commit}(f) - f(z), g_2)$$

$$?=$$

$$e(\mathrm{commit}(q), (s - z)g_2)$$

other openings

DARK (UO group)

$\mathrm{even}(f_i)(z), \mathrm{odd}(f_i)(z)$

Bulletproof (discrete log)

$\langle \mathrm{coeff}(f), \mathrm{powers}(x) \rangle$

recent developments

novel constructions

- lattice-based polynomial commitment
- Jacobian groups with unknown order
- sparse polynomial commitments

part 1—context

part 2—landscape

part 3—mechanics

part 4—gadgets

modularity

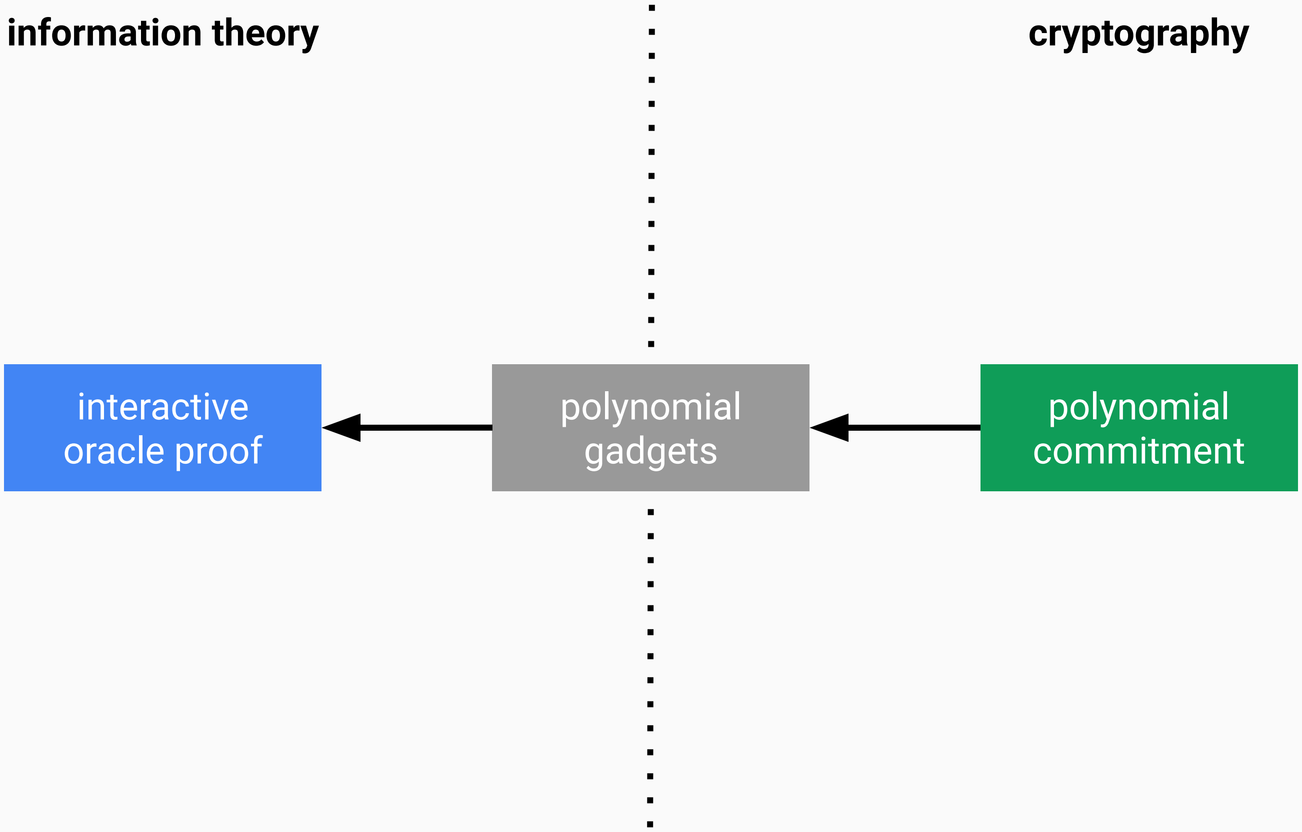
information theory

cryptography

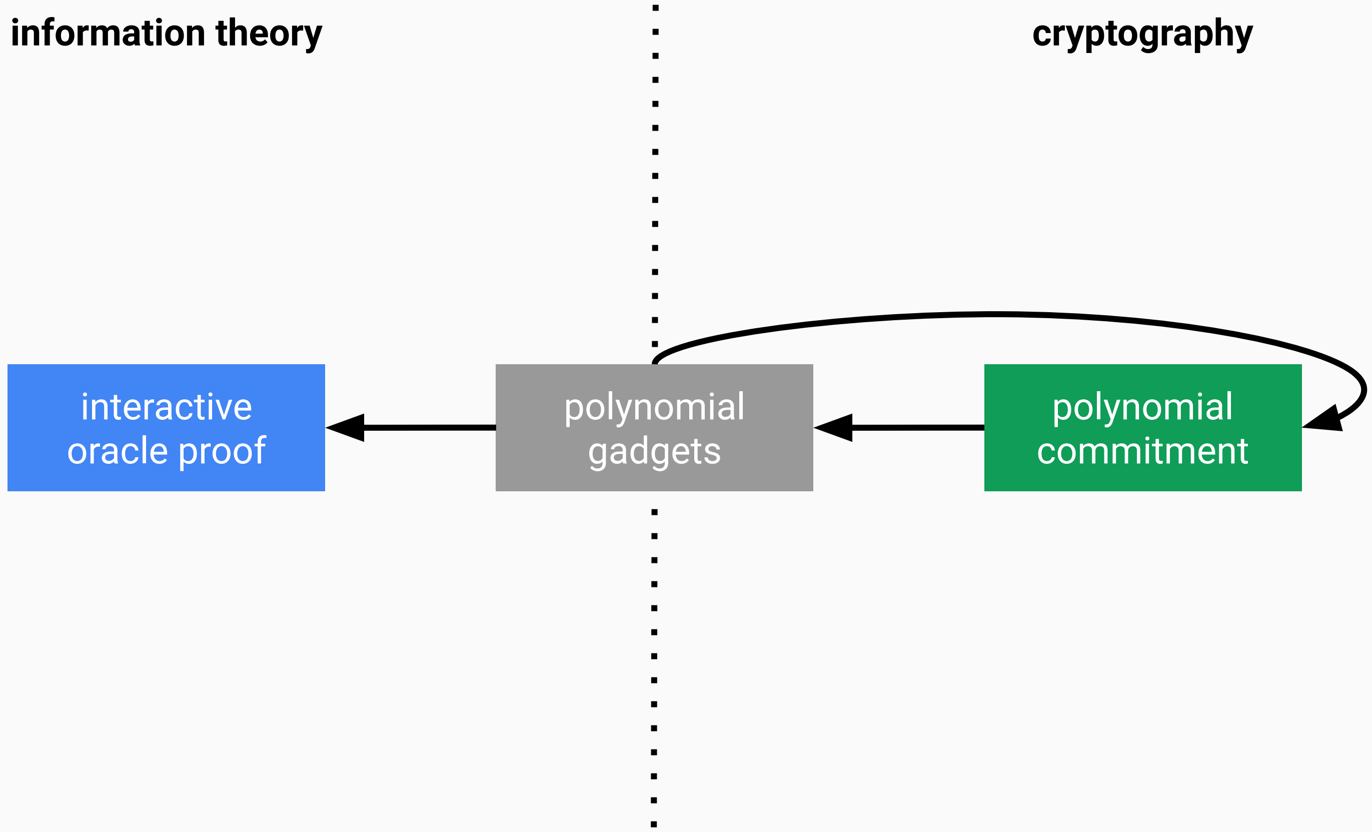
interactive
oracle proof

polynomial
commitment

modularity


information theory
cryptography
interactive oracle proof
polynomial gadgets
polynomial commitment

modularity


information theory
cryptography
interactive oracle proof
polynomial gadgets
polynomial commitment

testing polynomial identities

fundamental theorem of algebra

$\mathbf{f_1, f_2}$ low-degree polynomials

$\mathbf{f_1 = f_2}$ with high probability

$\Leftrightarrow$

$\mathbf{f_1(z) = f_2(z)}$ at random point z

Schwartz–Zippel lemma

$\mathbf{f_1(X), \ldots, f_k(X)}$ low-degree polynomials

$\mathbf{G(X_1, \ldots, X_K, Y)}$ low-degree

$\mathbf{G(f_1, \ldots, f_n, Y) = 0}$ with high probability

$\Leftrightarrow$

$\mathbf{G(f_1, \ldots, f_n, Y)|_{X=z}}$ at random point z

basic tricks

	trick
range	$(f \mathbin{//} Z_s) * Z_s$

range check

$$\xrightarrow{\text{commitments to } \mathbf{f}, \ \mathbf{f} \ /\!/ \ \mathbf{Z_s}}$$

range check

commitments to $\mathbf{f}$, $\mathbf{f}$ // $\mathbf{Z_S}$

$\mathbf{z}$

range check

commitments to **f**, **f** // $\mathbf{Z_S}$ →

← **z**

openings of **f**, **f** // $\mathbf{Z_S}$ at **z** →

range check

commitments to **f, f // Z_S**

z

openings of **f, f // Z_S** at **z**

check that **f(z) = (f // Z_S)(z)*Z_S(z)**

basic tricks

	trick
range	$(f \,//\, Z_S)*Z_S$
multi-point opening	$(f \,//\, Z_S)*Z_S + f \,\%\, Z_S$

multi-point opening

in the Lagrange basis
(evaluations of **f** on **S**)

commitments to **f**, **f // Z_S** plus **f % Z_S**

multi-point opening

in the Lagrange basis
(evaluations of **f** on **S**)

commitments to **f**, $\mathbf{f} \mathbin{/\!/} \mathbf{Z_S}$ plus $\mathbf{f} \mathbin{\%} \mathbf{Z_S}$

z

multi-point opening

in the Lagrange basis
(evaluations of **f** on **S**)

commitments to $\mathbf{f}$, $\mathbf{f} \mathbin{//} \mathbf{Z_S}$ plus $\mathbf{f} \mathbin{\%} \mathbf{Z_S}$

$\mathbf{z}$

openings of $\mathbf{f}$, $\mathbf{f} \mathbin{//} \mathbf{Z_S}$ at $\mathbf{z}$

multi-point opening

in the Lagrange basis
(evaluations of **f** on **S**)

commitments to **f**, **f //** $\mathbf{Z_s}$ plus **f %** $\mathbf{Z_s}$

z

openings of **f**, **f //** $\mathbf{Z_s}$ at **z**

check that **f(z) = (f //** $\mathbf{Z_s}$**)(z)***$\mathbf{Z_s}$**(z) + (f %** $\mathbf{Z_s}$**)(z)**

basic tricks

	trick
range	$(f \mathbin{//} Z_S)*Z_S$
multi-point opening	$(f \mathbin{//} Z_S)*Z_S + f \% Z_S$
multi-polynomial opening	$Y^0 f_0 + \ldots + Y^k f_k$

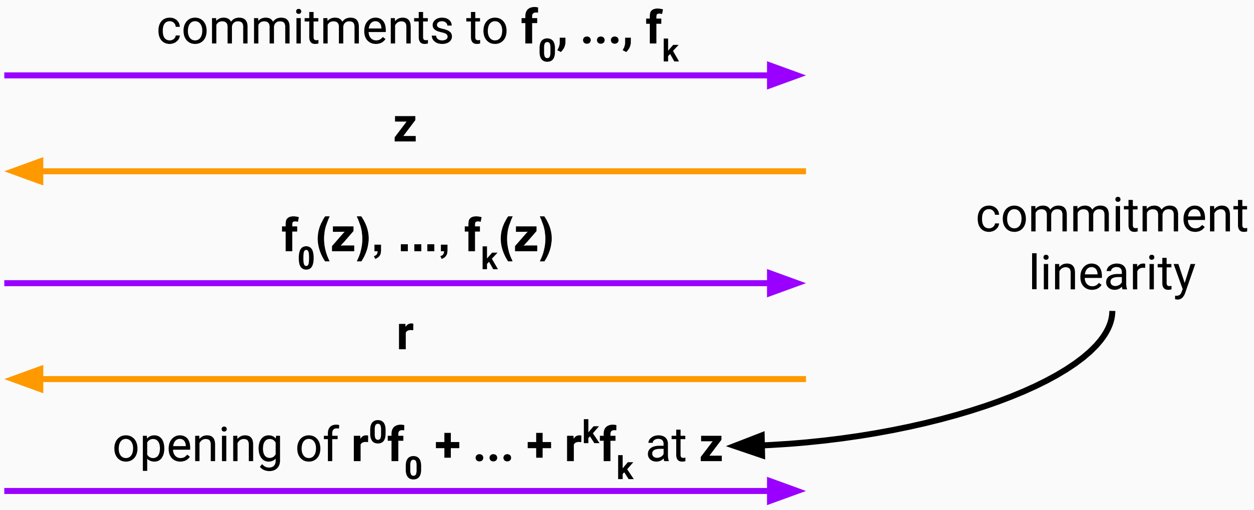
multi-point opening

commitments to $\mathbf{f_0, \dots, f_k}$

$\mathbf{z}$

$\mathbf{f_0(z), \dots, f_k(z)}$

multi-point opening



multi-point opening

commitments to $\mathbf{f_0, \ldots, f_k}$ $\longrightarrow$

$\mathbf{z}$ $\longleftarrow$

$\mathbf{f_0(z), \ldots, f_k(z)}$ $\longrightarrow$

$\mathbf{r}$ $\longleftarrow$

opening of $\mathbf{r^0 f_0 + \ldots + r^k f_k}$ at $\mathbf{z}$ $\longrightarrow$

commitment linearity

check that $\mathbf{(r^0 f_0 + \ldots + r^k f_k)(z) = r^0 f_0(z) + \ldots + r^k f_k(z)}$

basic tricks

	trick
range	$(f \,//\, Z_S)^*Z_S$
multi-point opening	$(f \,//\, Z_S)^*Z_S + f \,\%\, Z_S$
multi-polynomial opening	$Y^0f_0 + \ldots + Y^kf_k$
multi-{point, polynomial}	see here
degree bound	$X^{N-d}f(X)$

side note—Lagrange basis

monomial and Lagrange bases

$(a_0, \ldots, a_d)$

vector

monomial and Lagrange bases

$(a_0, \ldots, a_d)$

vector

$a_0X^0 + \ldots + a_dX^d$

monomial basis

monomial and Lagrange bases

$$(a_0, \dots, a_d)$$

vector

$$a_0X^0 + \dots + a_dX^d$$

monomial basis

$$a_0L_0(X) + \dots + a_dL_d(X)$$

Lagrange basis

monomial and Lagrange bases

$(a_0, \ldots, a_d)$

vector

$a_0X^0 + \ldots + a_dX^d$

monomial basis

$a_0L_0(X) + \ldots + a_dL_d(X)$

Lagrange basis

FFT

inverse FFT

barycentric formula

evaluation on **i**'th root of unity

$$f(z) = \frac{z^n - 1}{n} \sum_{i=1}^{n} \frac{f(\omega^i)}{(z - \omega^i)\omega^i}$$

Montgomery batch inversion

a_0
a_1
a_2
a_3
a_4
a_5

Montgomery batch inversion

a_0	a_0
a_1	a_0a_1
a_2	$a_0a_1a_2$
a_3	$a_0a_1a_2a_3$
a_4	$a_0a_1a_2a_3a_4$
a_5	$\boxed{a_0a_1a_2a_3a_4a_5}$

Montgomery batch inversion

$$
\begin{array}{lll}
a_0 & & a_0 \\
a_1 & & a_0a_1 \\
a_2 & & a_0a_1a_2 \\
a_3 & & a_0a_1a_2a_3 \\
a_4 & & a_0a_1a_2a_3a_4 \\
a_5 & & \boxed{a_0a_1a_2a_3a_4a_5} \quad \boxed{(a_0a_1a_2a_3a_4a_5)^{-1}}
\end{array}
$$

Montgomery batch inversion

a_0
a_1
a_2
a_3
a_4
a_5

a_0
a_0a_1
$a_0a_1a_2$
$a_0a_1a_2a_3$
$a_0a_1a_2a_3a_4$
$\boxed{a_0a_1a_2a_3a_4a_5}$

$(a_0)^{-1}$
$(a_0a_1)^{-1}$
$(a_0a_1a_2)^{-1}$
$(a_0a_1a_2a_3)^{-1}$
$(a_0a_1a_2a_3a_4)^{-1}$
$\boxed{(a_0a_1a_2a_3a_4a_5)^{-1}}$

Montgomery batch inversion

a_0	a_0	$(a_0)^{-1}$	$(a_0)^{-1}$
a_1	a_0a_1	$(a_0a_1)^{-1}$	$(a_1)^{-1}$
a_2	$a_0a_1a_2$	$(a_0a_1a_2)^{-1}$	$(a_2)^{-1}$
a_3	$a_0a_1a_2a_3$	$(a_0a_1a_2a_3)^{-1}$	$(a_3)^{-1}$
a_4	$a_0a_1a_2a_3a_4$	$(a_0a_1a_2a_3a_4)^{-1}$	$(a_4)^{-1}$
a_5	$\boxed{a_0a_1a_2a_3a_4a_5}$	$\boxed{(a_0a_1a_2a_3a_4a_5)^{-1}}$	$(a_5)^{-1}$

/side note—Lagrange basis

more tricks

	Lagrange basis	monomial basis
encode	$a_0L_0(X) + \ldots + a_dL_d(X)$	$a_0X^0 + \ldots + a_dX^d$

more tricks

	Lagrange basis	monomial basis
encode	$a_0L_0(X) + \ldots + a_dL_d(X)$	$a_0X^0 + \ldots + a_dX^d$
query	$f(w^i)$	$f_L(X) + a_iX^i + X^{i+1}f_R(X)$

more tricks

	Lagrange basis	monomial basis
encode	$a_0L_0(X) + \ldots + a_dL_d(X)$	$a_0X^0 + \ldots + a_dX^d$
query	$f(w^i)$	$f_L(X) + a_iX^i + X^{i+1}f_R(X)$
shift	$f(w^iX)$	$X^if(X)$

more tricks

	Lagrange basis	monomial basis
encode	$a_0L_0(X) + \ldots + a_dL_d(X)$	$a_0X^0 + \ldots + a_dX^d$
query	$f(w^i)$	$f_L(X) + a_iX^i + X^{i+1}f_R(X)$
shift	$f(w^iX)$	$X^if(X)$
sum	$g(wX) = f(X) + g(X)$	$f(1)$

sum argument

$q = (\mathrm{shift}(g) - g - f)\ //\ Z_S$

commitments to **f, g, q**

sum argument

$\mathbf{q = (shift(g) - g - f)\ //\ Z_S}$

commitments to **f, g, q**

z

sum argument

q = (shift(g) - g - f) // Z_S

commitments to **f, g, q**

z

openings of **f, g, shift(g), q** at **z**

sum argument

q = (shift(g) - g - f) // Z_S

commitments to **f, g, q**

z

openings of **f, g, shift(g), q** at **z**

shift(g)(z)

check **g(wz) - g(z) - f(z) = q(z)Z_S(z)**

more tricks

	Lagrange basis	monomial basis
encode	$a_0L_0(X) + \ldots + a_dL_d(X)$	$a_0X^0 + \ldots + a_dX^d$
query	$f(w^i)$	$f_L(X) + a_iX^i + X^{i+1}f_R(X)$
shift	$f(w^iX)$	$X^if(X)$
sum	$g(wX) = f(X) + g(X)$	$f(1)$

sum check alternative

$|S|^{-1}(f(X) \% Z_S(X))|_{X=0}$

more tricks

	Lagrange basis	monomial basis
encode	$a_0L_0(X) + \dots + a_dL_d(X)$	$a_0X^0 + \dots + a_dX^d$
query	$f(w^i)$	$f_L(X) + a_iX^i + X^{i+1}f_R(X)$
shift	$f(w^iX)$	$X^if(X)$
sum	$g(wX) = f(X) + g(X)$	$f(1)$
grand product	$g(wX) = f(X)g(X)$	see Sonic appendix B

more tricks

	Lagrange basis	monomial basis
encode	$a_0L_0(X) + \ldots + a_dL_d(X)$	$a_0X^0 + \ldots + a_dX^d$
query	$f(w^i)$	$f_L(X) + a_iX^i + X^{i+1}f_R(X)$
shift	$f(w^iX)$	$X^if(X)$
sum	$g(wX) = f(X) + g(X)$	$f(1)$
grand product	$g(wX) = f(X)g(X)$	see Sonic appendix B
permutation	$f(X) + Y\sigma(X) + Z$ and $f(X) + YX + Z$ grand products	see Sonic appendix A

permutation argument

$\boldsymbol{\sigma}$**:** $(\mathbf{a}_i) \rightarrow (\mathbf{a}_j)$ is a permutation

permutation argument

$$\boldsymbol{\sigma}\mathbf{: (a_i)} \rightarrow \mathbf{(a_j)} \text{ is a permutation}$$

$$\Leftrightarrow$$

$$\mathbf{a_i = a_j} \text{ whenever } \mathbf{i = \boldsymbol{\sigma}(j)}$$

permutation argument

$$\boldsymbol{\sigma}\mathbf{:}\ (\mathbf{a_i}) \to (\mathbf{a_j}) \text{ is a permutation}$$

$$\Leftrightarrow$$

$$\mathbf{a_i = a_j} \text{ whenever } \mathbf{i = \boldsymbol{\sigma}(j)}$$

$$\Leftrightarrow$$

$$\mathbf{a_i + i{*}X = a_j + \boldsymbol{\sigma}(j){*}X} \text{ whenever } \mathbf{i = \boldsymbol{\sigma}(j)}$$

permutation argument

$\boldsymbol{\sigma}$**:** $(\mathbf{a_i}) \rightarrow (\mathbf{a_j})$ is a permutation

$\Leftrightarrow$

$\mathbf{a_i = a_j}$ whenever $\mathbf{i = \sigma(j)}$

$\Leftrightarrow$

$\mathbf{a_i + i{*}X = a_j + \sigma(j){*}X}$ whenever $\mathbf{i = \sigma(j)}$

$\Leftrightarrow$

$\mathbf{\{a_i + i{*}X\} = \{a_j + \sigma(j){*}X\}}$ as multisets

permutation argument

$\boldsymbol{\sigma}$**: $(\mathbf{a}_i) \rightarrow (\mathbf{a}_j)$** is a permutation

$\Leftrightarrow$

$\mathbf{a}_i = \mathbf{a}_j$ whenever $\mathbf{i = \sigma(j)}$

$\Leftrightarrow$

$\mathbf{a}_i + \mathbf{i*X} = \mathbf{a}_j + \boldsymbol{\sigma}\mathbf{(j)*X}$ whenever $\mathbf{i = \sigma(j)}$

$\Leftrightarrow$

$\{\mathbf{a}_i + \mathbf{i*X}\} = \{\mathbf{a}_j + \boldsymbol{\sigma}\mathbf{(j)*X}\}$ as multisets

$\Leftrightarrow$

$\mathbf{product}_i(\mathbf{a}_i + \mathbf{i*X + Y}) = \mathbf{product}_j(\mathbf{a}_j + \boldsymbol{\sigma}\mathbf{(j)*X + Y})$ as polynomials in **X, Y**

permutation argument

$$\boldsymbol{\sigma}\textbf{: (}\mathbf{a_i}\textbf{)} \rightarrow \textbf{(}\mathbf{a_j}\textbf{)} \text{ is a permutation}$$

$$\Leftrightarrow$$

$$\mathbf{a_i = a_j} \text{ whenever } \mathbf{i = \sigma(j)}$$

$$\Leftrightarrow$$

$$\mathbf{a_i + i{*}X = a_j + \sigma(j){*}X} \text{ whenever } \mathbf{i = \sigma(j)}$$

$$\Leftrightarrow$$

$$\mathbf{\{a_i + i{*}X\} = \{a_j + \sigma(j){*}X\}} \text{ as multisets}$$

$$\Leftrightarrow$$

$$\mathbf{product_i(a_i + i{*}X + Y) = product_j(a_j + \sigma(j){*}X + Y)} \text{ as polynomials in } \mathbf{X, Y}$$

$$\Leftrightarrow$$

$$\mathbf{product_i(a_i + i{*}r_1 + r_2) = product_j(a_j + \sigma(j){*}r_1 + r_2)} \text{ for random challenges } \mathbf{r_1, r_2}$$

even more tricks

Lagrange superpower

	Lagrange basis trick
Hadamard product	$f_1(X)f_2(X)$

even more tricks

Lagrange superpower

	Lagrange basis trick
Hadamard product	$\mathbf{f_1(X)f_2(X)}$
inner product	sum over $\mathbf{f_1(X)f_2(X)}$

even more tricks

Lagrange superpower

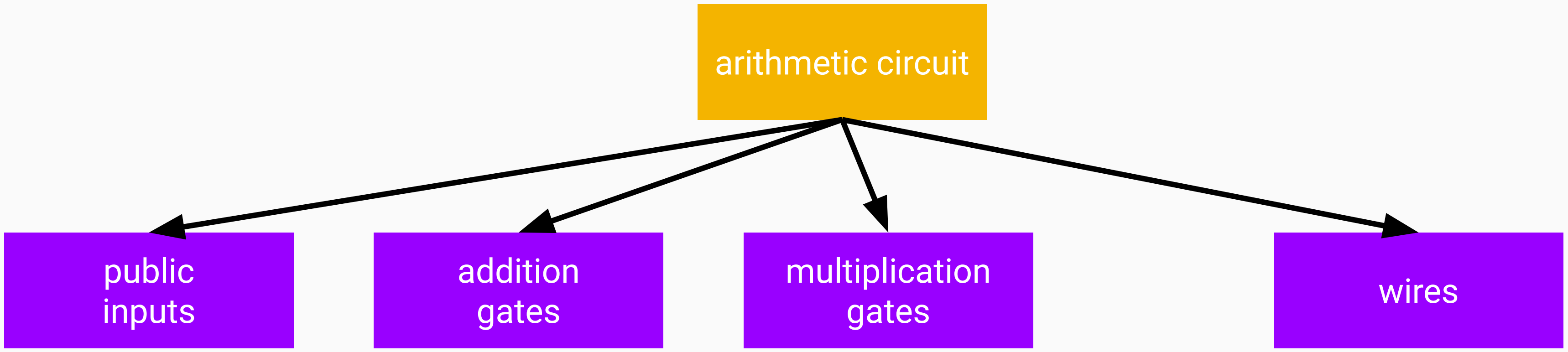
	Lagrange basis trick
Hadamard product	$\mathbf{f_1(X)f_2(X)}$
inner product	sum over $\mathbf{f_1(X)f_2(X)}$
sparse matrix multiplication	two sum cheks; see [here](#)

even more tricks

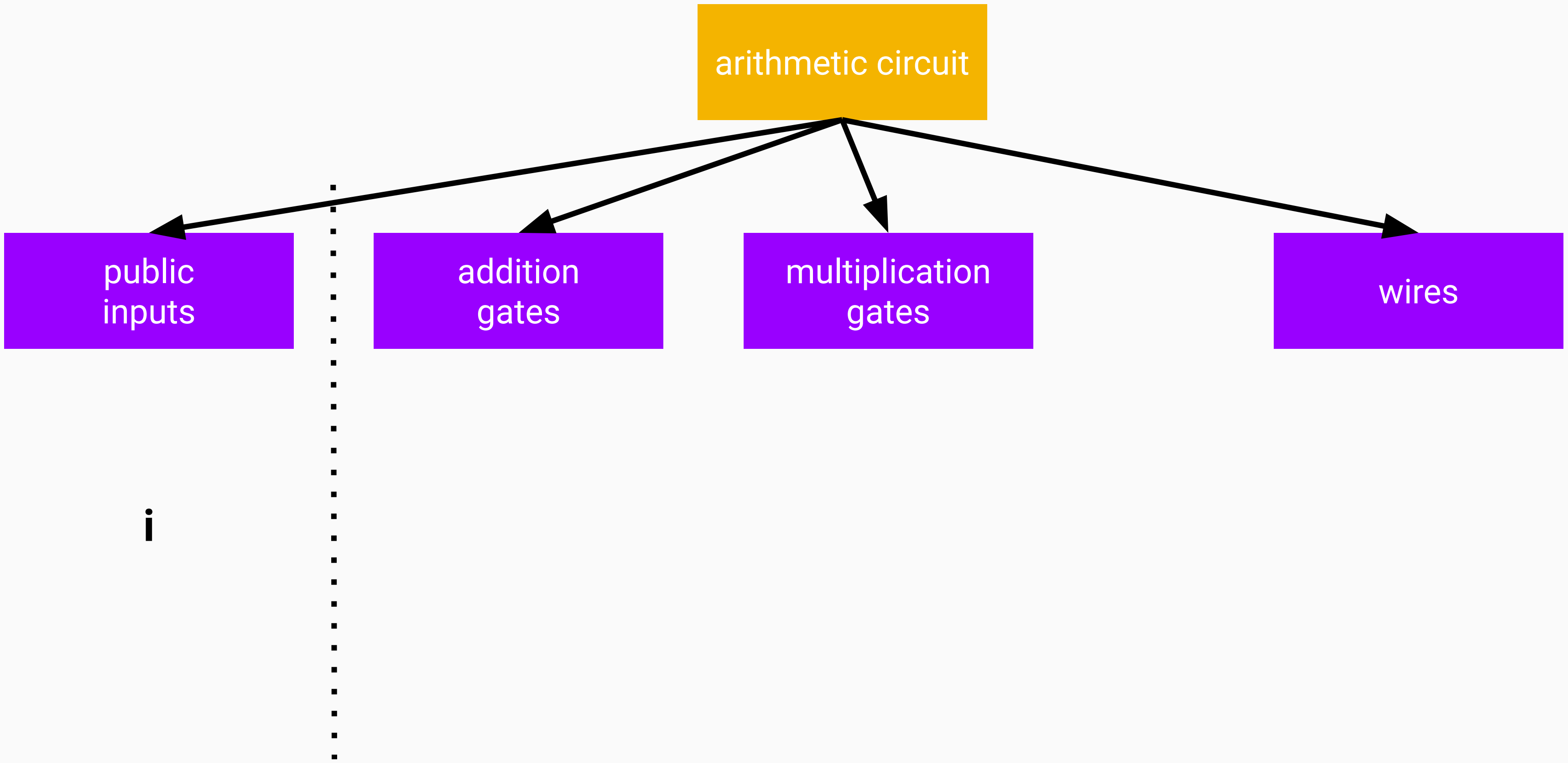
Lagrange superpower

	Lagrange basis trick
Hadamard product	$\mathbf{f_1(X)f_2(X)}$
inner product	sum over $\mathbf{f_1(X)f_2(X)}$
sparse matrix multiplication	two sum cheks; see here
range checks	see Aztec research
RAM read and write	see Aztec research

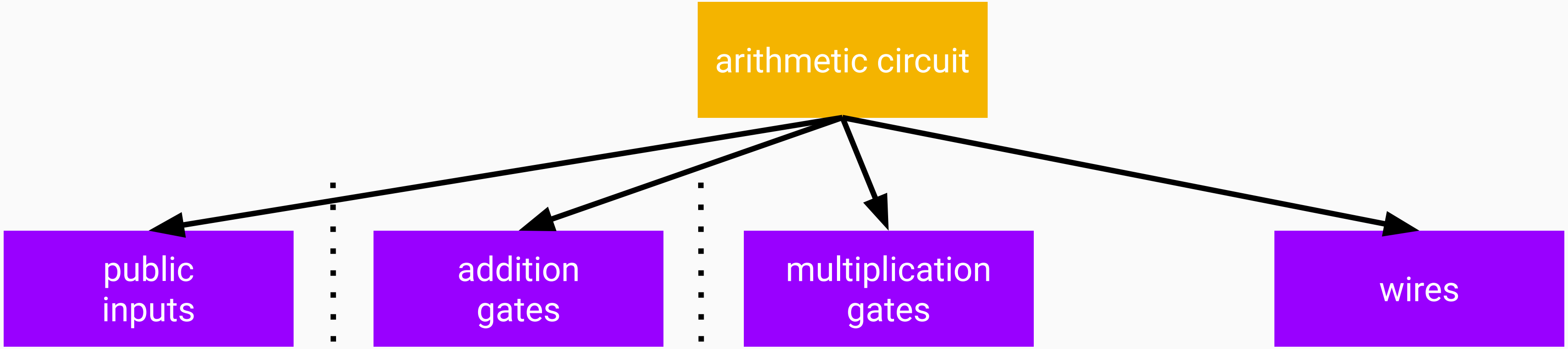
PLONK constraint system



PLONK constraint system



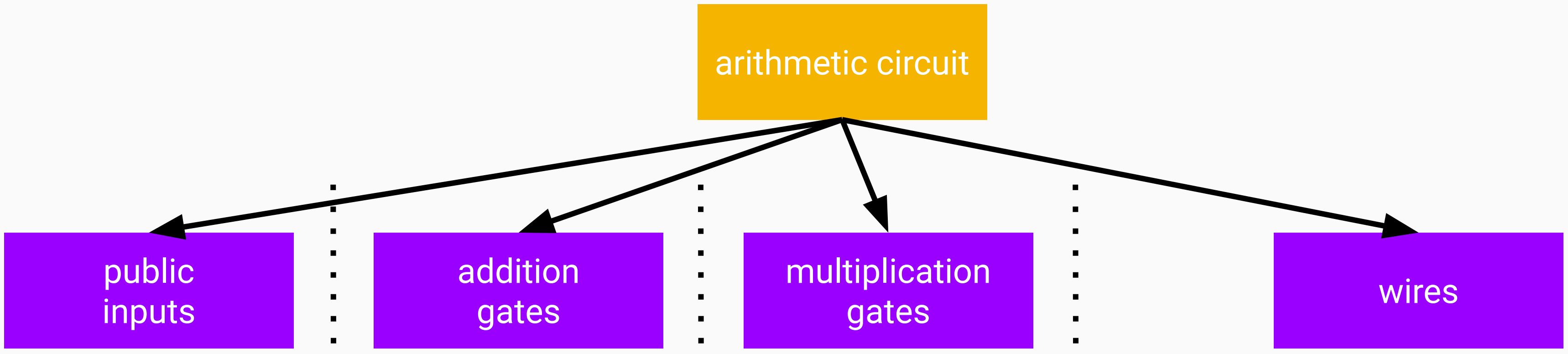
PLONK constraint system



i

$a_L + a_R = a_O$

PLONK constraint system

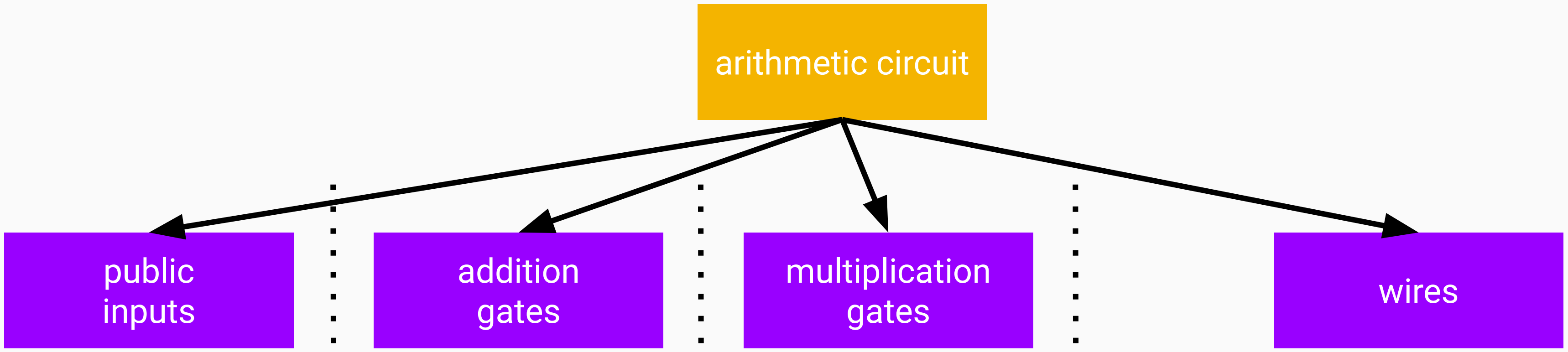


i

$a_L + a_R = a_O$

$m_L m_R = m_O$

PLONK constraint system



i

$a_L + a_R = a_O$

$m_L m_R = m_O$

```
w = concat(
    i,              # input wires
    aL, aR, aO,     #  ADD wires
    mL, mR, mO,     #  MUL wires
)

w = permute(w, σ)
```

thank you :)