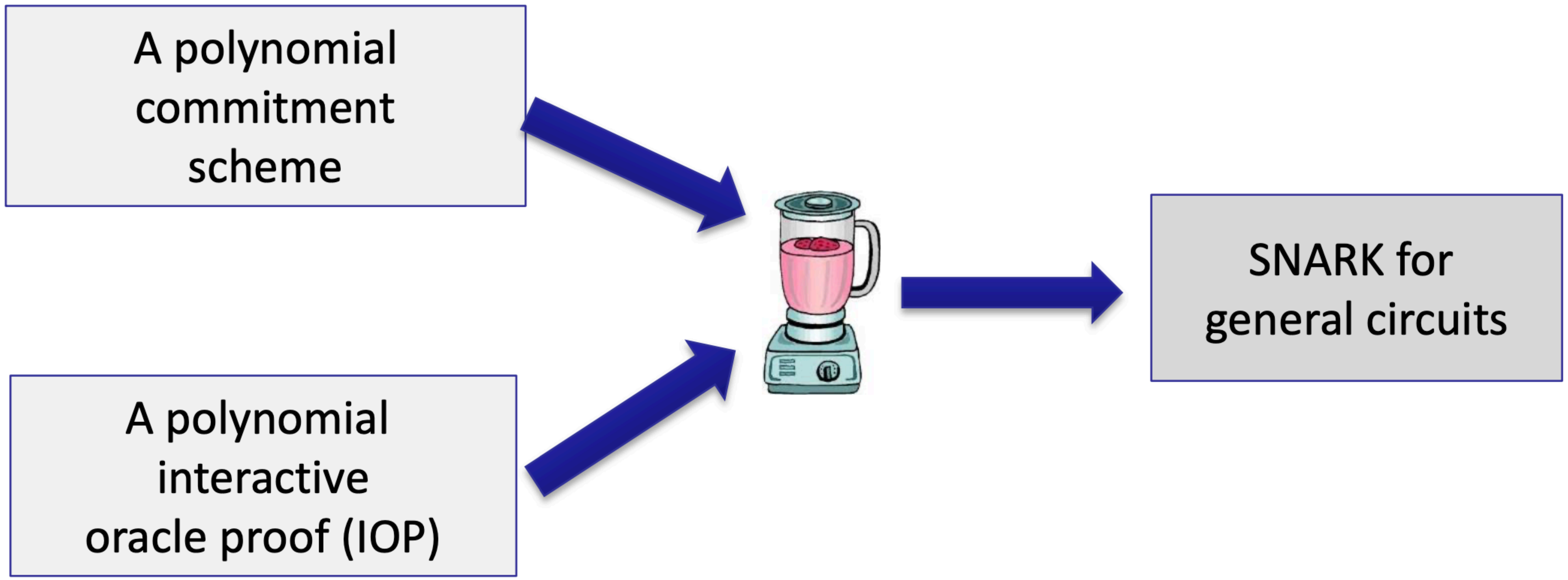
Discrete log based PCS

ZK-School Beginner class
Mentor: Inseon Yu

Outline

- Pairing based PCS: KZG
- Discrete logarithm based PCS: Bulletproofs
- Taxonomy of SNARKs

Recall: modern SNARK construction

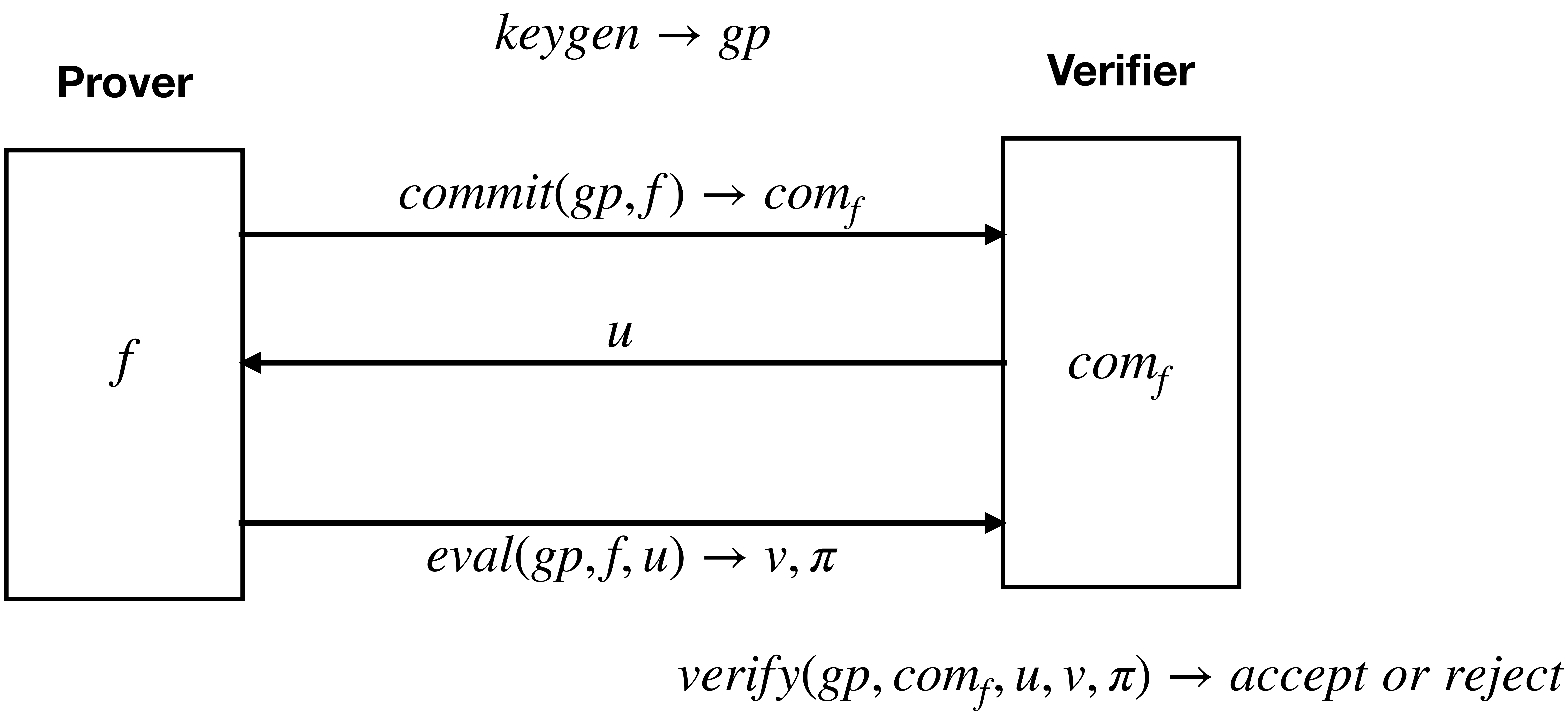


Recall: Polynomial Commitment Scheme

PCS construction

- $keygen \rightarrow gp$
- $commit(gp, f) \rightarrow com_f$
- $eval(gp, f, u) \rightarrow v, \pi$
- $verify(gp, com_f, u, v, \pi) \rightarrow accept\ or\ reject$

Recall: Polynomial Commitment Scheme



Outline

- **Pairing based PCS: KZG**
- Discrete logarithm based PCS: Bulletproofs
- Taxonomy of SNARKs

Recall: Pairing based PCS

- **Pairing :**

  $e : G1 \times G2 \rightarrow GT$

- **Bilinearity :**

  $e(aP, bQ) = e(P, bQ)^a = e(P, Q)^{ab} = e(P, aQ)^b = e(bP, aQ)$

- **CDH vs DDH**

  CDH(Computational Diffie-Hellman) : Solving the exact value of abG from aG, bG

  DDH(Decisional Diffie-Hellman) : Determining if abG is valid

KZG [Kate-Zaverucha-Goldberg' 2010]

Setup

- Bilinear group $p,\, G \in \mathbb{G},\, \mathbb{G}_T,\, e$
- Univariate polynomials $F = \mathbb{F}_p^{(\leq d)}[X]$
- Keygen:
  - Sample random $\tau \in \mathbb{F}_p$
  - $gp = (G, \tau G, \tau^2 G, \ldots, \tau^d G)$
  - delete τ

KZG [Kate-Zaverucha-Goldberg' 2010]

Commit

- $gp = (G, \tau G, \tau^2 G, \ldots, \tau^d G)$
- $commit(gp, f) \to com_f:$
  - $f(x) = f_0 + f_1 x + f_2 x^2 + \ldots + f_d x^d$
  - $com_f = f(\tau) \cdot G = (f_0 + f_1 \tau + \ldots + f_d \tau^d) \cdot G$

$$= f_0 \cdot G + f_1 \cdot \tau G + \ldots + f_d \cdot \tau^d G$$

KZG [Kate-Zaverucha-Goldberg' 2010]

Evaluation

- $gp = (G,\ \tau G,\ \tau^2 G,\ \ldots\ ,\ \tau^d G)$
- $eval(gp, com_f, u) \rightarrow v, \pi:$
  - $f(x) - v = (x - u)q(x)$
  - compute $q(x)$ and $\pi = q(\tau)\ G$

KZG [Kate-Zaverucha-Goldberg’ 2010]

Verification

- $f(x) - f(u) = (x - u)\ q(x)$

- Honest prover: $com_f = f(\tau) \cdot G,\ \pi = q(\tau) \cdot G,\ v = f(u)$

- Check the point at τ : $(f(\tau) - f(u)) \cdot G = ((\tau - u)\ q(\tau)) \cdot G$ (X)
  - only know $(\tau - u) \cdot G,\ q(\tau) \cdot G$

- Pairing :
  - $e((com_f - v) \cdot G,\ G) = e((f(\tau) - f(u)) \cdot G,\ G) = e(G,\ G)^{f(\tau) - f(u)}$
  - $e((\tau - u) \cdot G,\ \pi) = e((\tau - u) \cdot G,\ q(t) \cdot G) = e(G,\ G)^{(\tau - u) q(\tau)}$

  → $e(G, G)^{f(\tau) - f(u)} = e(G, G)^{(\tau - u)\ q(\tau)}$

KZG [Kate-Zaverucha-Goldberg' 2010]

$$gp = (G, \tau G, \tau^2 G, \ldots, \tau^d G)$$

Prover

Verifier

$f(x)$

$com_f = f(\tau) \cdot G$

u

$v, \pi = q(\tau) \cdot G$

com_f

$$f(x) - f(u) = (x - u)\, q(x)$$

$$e(G, G)^{f(\tau) - f(u)} = e(G, G)^{(\tau - u)\, q(\tau)}$$

Ceremony

- A distributed generation of gp s.t. no one can reconstruct the trapdoor if at least one of the participants is honest and discards their secrets
- $gp = (\tau G, \tau^2 G, \ldots, \tau^d G) = (G_1, G_2, \ldots, G_d)$
- Sample random s, update with secret τ, s :
  $gp' = (G', G_2', \ldots, G_d') = (sG_1, s^2 G_2, \ldots, s^d G_d) = (\tau s G, (\tau s)^2 G, \ldots, (\tau s)^d G)$
- Check the correctness of gp'
  1. The contributor knows s s.t. $G_1' = sG_1$
  2. gp' consist of consecutive powers $e(G_i', G_1') = e(G_{i+1}' G)$ and, $G_1' \neq 1$

Multivariate Polynomial Commitment

[Papamanthou-Shi-Tamassia'13]

Key idea: $f(x_1, \ldots, x_k) - f(u_1, \ldots, u_k) = \sum_{i=1}^{k} (x_i - u_i) q_i(\vec{x})$

- Keygen: compute gp as G raised to all possible monomials of $\tau_1, \tau_2, \ldots, \tau_k$
- Commit: $com_f = f(\tau_1, \tau_2, \ldots, \tau_k) \cdot G$
- Eval: $\pi_i = q_i(\vec{\tau}) \cdot G$

$\rightarrow O(\log n)$ proof size and verifier time

- Verify: $e((com_f - v) \cdot G,\, G) = \prod_{i=1}^{k} e((\tau - u) \cdot G, \pi_i)$

Achieving zero-knowledge[ZGKPP'2018]

- Plain KZG is not ZK. E.g., $com_f = f(\tau)\cdot G$ is deterministic

- Solution: masking with randomizer

  - Commit: $com_f = (f(\tau) + r\eta)\cdot G$

  - Eval: $f(x) + ry - f(u) = (x-u)(q(x) + r'y) + y(r - r'(x-u))$

    $\pi = (q(\tau) + r'\eta)\cdot G, (r - r'(\tau - u))\cdot G$

Batch opening: single polynomial

Prover wants to prove f at $u_1, \ldots, u_m$ for $m < d$

- Key idea:
  - Extrapolate $f(u_1), \ldots, f(u_m)$ to get $h(x)$
  - $f(x) - h(x) = \prod_{i=1}^{m} (x - u_i)\, q(x)$
  - $\pi = q(\tau) \cdot G$
  - $e((com_f - h(\tau)) \cdot G, G) = \prod_{i=1}^{k} e((\tau - u_i) \cdot G, \pi)$

Batch opening: multiple polynomials

Prover wants to prove $f_i(u_{i,j}) = v_{i,j} \ for \ i \in [n], j \in [m]$

- Key idea:
  - Extrapolate $f_i(u_1), \ldots, f_i(u_m)$ to get $h_i(x)$ for $i \in [n]$
  - $f_i(x) - h_i(x) = \prod_{j=1}^{m} (x - u_j)\ q_i(x)$
  - combine all $q_i(x)$ via a random linear combination

- Feist-Khovratovich (FK) algorithm (2020) :
  - If U is a multiplicative subgroup: $O(n \log n)$
  - Otherwise: $O(n \log^2 n)$

Pros and Cons of KZG

- **Pros:**

  - Commitment and Proof size: $O(1)$

  - Verifier time: $O(1)$ pairing

- **Cons:**

  - Trusted setup

Outline

- Pairing based PCS: KZG
- **Discrete logarithm based PCS: Bulletproofs**
- Taxonomy of SNARKs

Discrete log based PCS

- A group $\mathbb{G}$ has an alternative representation as the powers of the generator G : $\{G, G^2, G^3, \ldots, G^{p-1}\} := \{G, 2G, \ldots, (p-1)G\}$
- Discrete logarithm problem: given $y \in \mathbb{G}$, find x s.t. $x \cdot G = y$
- Discrete log assumption: DLP is computationally hard

Inner Product

- $a = (a_0, a_1, \ldots, a_{n-1})$, $b = (b_0, b_1, \ldots, b_{n-1})$
  Inner product $< a, b > = a_0 b_0 + a_1 b_1 + \ldots + a_{n-1} b_{n-1}$

- Given:
  $p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1}$
  $a = (a_0, a_1, \ldots, a_{n-1})$, $z = (1, z^1, z^2, \ldots, z^{n-1})$

- Inner product $< a, z >$ denotes $p(z)$

Pedersen commitment

- $G, H \in \mathbb{G}$

- $Commit(m; r) = [m]G + [r]H$

- Pedersen vector commitment with vector m, G:

  $[r]H + m_0G_0 + m_1G_1 + \ldots + m_{n-1}G_{n-1}$

  $= [r]H + < m, G >$

Bulletproofs

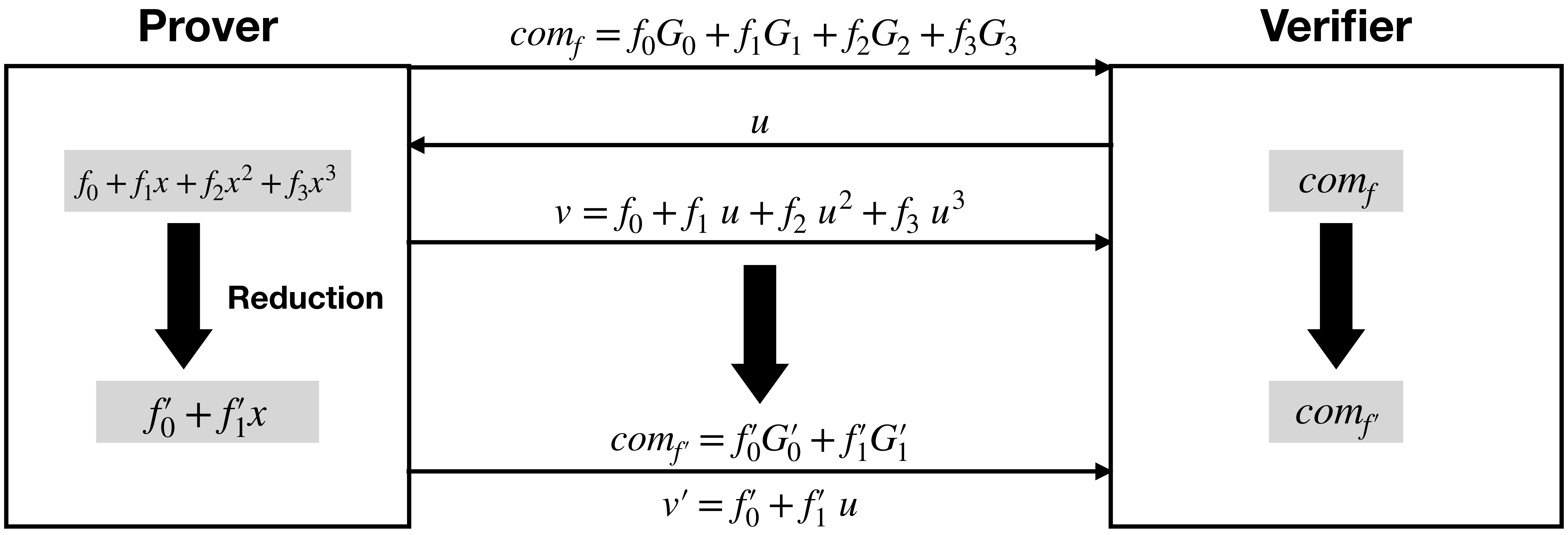
- BCCGP’16 : Proposed Inner Product Argument
- BBBPWM’18 : Optimize IPA → Bulletproofs
- Can be generalized to proofs for a general arithmetic circuits and have special protocol like range proofs

Bulletproofs

- Transparent setup: sample random $gp = (G_0, G_1, \ldots, G_d)$ in $\mathbb{G}$
- Commit: $f(x) = f_0 + f_1 x + f_2 x^2 + \ldots + f_d x^d$

$$com_f = f_0 G_0 + f_1 G_1 + \ldots + f_d G_d$$

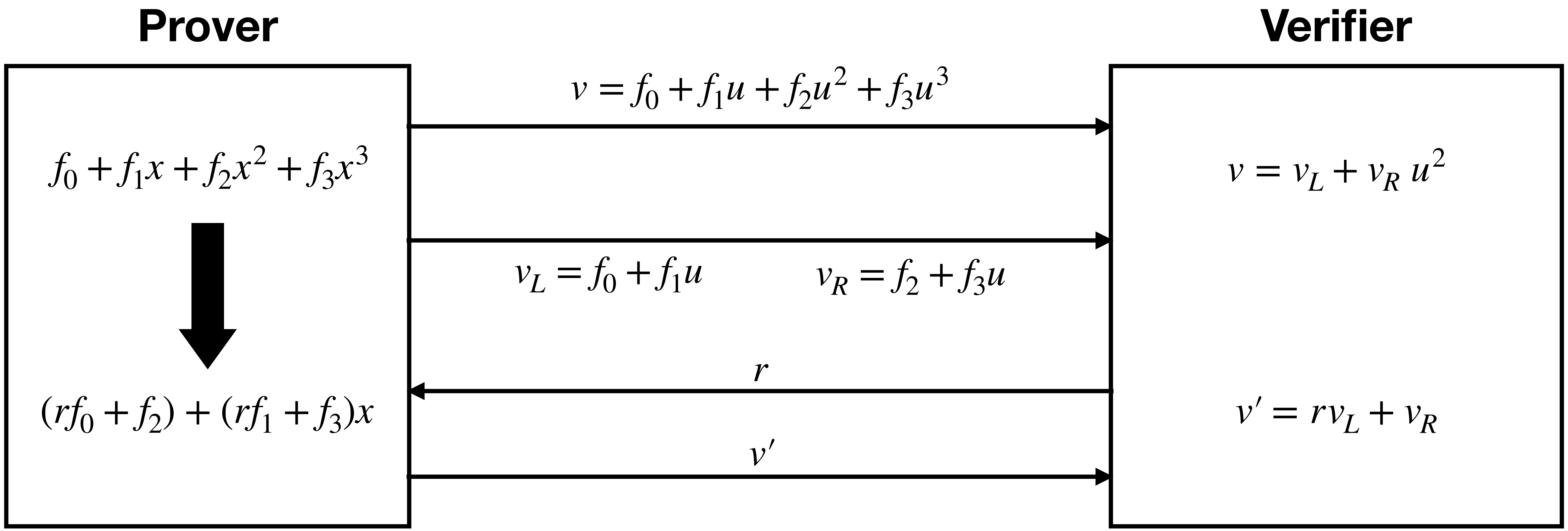
High-level idea



Bulletproofs

$$gp = (G_0, G_1, G_2, G_3)$$

$$com_f = f_0 G_0 + f_1 G_1 + f_2 G_2 + f_3 G_3$$



Combine polynomials via random linear combination

Bulletproofs

$$gp = (G_0, G_1, G_2, G_3)$$

$$com_f = f_0 G_0 + f_1 G_1 + f_2 G_2 + f_3 G_3$$

Prover

$f_0 + f_1 x + f_2 x^2 + f_3 x^3$

$(rf_0 + f_2) + (rf_1 + f_3)x$

Verifier

$v = v_L + v_R\, u^2$

$v' = rv_L + v_R$

$gp' = (r^{-1}G_0 + G_2,\ r^{-1}G_1 + G_3)$

$com' = rL + com_f + r^{-1}R$

Prover → Verifier: $v = f_0 + f_1 u + f_2 u^2 + f_3 u^3$

Prover → Verifier: $L = f_0 G_2 + f_1 G_3$ $\quad R = f_2 G_0 + f_3 G_1$, $\quad v_L = f_0 + f_1 u$ $\quad v_R = f_2 + f_3 u$

Verifier → Prover: r

Prover → Verifier: v'

Compute new commitment via L, R

Bulletproofs

- $com_f = f_0G_0 + f_1G_1 + f_2G_2 + f_3G_3$, $L = f_0G_2 + f_1G_3$, $R = f_2G_0 + f_3G_1$
- $com_{f'} = rL + com_f + r^{-1}R$

$$= (f_0 + r^{-1}f_2)G_0 + (rf_0 + f_2)G_2 \ + \ (f_1 + r^{-1}f_2)G_1 + (rf_1 + f_3)G_3$$

$$= \boxed{(rf_0 + f_2)}(r^{-1}G_0 + G_2) + \boxed{(rf_1 + f_3)}(r^{-1}G_1 + G_3)$$

- $gp' = (r^{-1}G_0 + G_2, r^{-1}G_1 + G_3)$

Bulletproofs

- **Eval**
  1. Compute L, R, v_L, v_R
  2. Receive r from verifier, reduce f to f' of degree $d/2$
  3. Update the bases gp'

- **Verify**
  1. Check $v = v_L + v_R \, u^{d/2}$
  2. Generate r randomly
  3. Update $com' = rL + com_f + r^{-1}R, \;\; gp', \;\; v' = rv_L + v_R$

Bulletproofs

- **Keygen:** $O(n)$, transparent setup
- **Eval:** $O(n)$ group exponentiations

  ($\rightarrow$ Non-Interactive via Fiat Shamir)
- **Proof size:** $O(\log n)$
- **Verifier time:** $O(n)$

Pros and Cons of Bulletproofs

- **Pros:**

  - Transparent setup

  - Proof is relatively short among transparent SNARKs

- **Cons:**

  - Slow verifier time

Improvements

- **Hyrax** [Wahby-Tzialla-shelat-Thaler-Walfish'18]

  - Improves the verifier time to $O(\sqrt{n})$ by representing the coefficients as a 2-D matrix

  - Proof size: $O(\sqrt{n})$

- **Dory** [Lee'2021]

  - Improving verifier time to $O(\log n)$

  - Key idea: delegating the structured verifier computation to the prover using **Inner pairing product arguments** [BMMTV'2021]

  - Also improves the prover time to $O(\sqrt{n})$ exponentiations plus $O(n)$field operations

Improvements

- **DARK** [Bünz-Fisch-Szepieniec'20]

  - Achieves O log d proof size and verifier time

  - Group of unknown order

Summary

Scheme	Prover	Proof size	Verifier	Trusted setup	Crypto primitive
KZG	$O(n)$	$O(1)$	$O(1)$	O	Pairing
Bulletproofs	$O(n)$	$O(\log n)$	$O(n)$	X	Discrete-log
Hyrax	$O(n)$	$O(\sqrt{n})$	$O(\sqrt{n})$	X	Discrete-log
Dory	$O(n)$	$O(\log n)$	$O(\log n)$	X	Pairing
Dark	$O(n)$	$O(\log n)$	$O(\log n)$	X	Unknown order group

Outline

- Pairing based PCS: KZG
- Discrete logarithm based PCS: Bulletproofs
- **Taxonomy of SNARKs**

Highlights of SNARK Taxonomy: Transparent SNARKs

- [Any polynomial IOP] + **IPA/bulletproofs**
  - Ex: Halo2 (ZCash)
  - **Pros:** Shortest proofs among transparent SNARKs
  - **Cons:** Slow verifier time

- [Any polynomial IOP] + **FRI**
  - Ex: STARK, Fractal, Aurora, Virgo, Ligero++
  - **Pros:** Shortest proofs amongst plausibly post-quantum SNARKs.
  - **Cons:** Proofs are large (100s of KBs depending on security)

- **MIPs and IPs** + [fast-prover polynomial commitments]
  - Ex: Spartan, Brakedown, Orion, Orion+
  - **Pros:** Fastest P in the literature, plausibly post-quantum + transparent if polynomial commitment is
  - **Cons:** Bigger proofs than others above.

Highlights of SNARK Taxonomy: Non-transparent SNARKs

- **Linear PCP based**
  - Ex: Groth16
  - **Pros**: Shortest proofs (3 group elements), fastest V
  - **Cons**: Circuit-specific trusted setup, slow and space-intensive P, not post- quantum

- **Constant-round PIOP + KZG**
  - Ex: Marlin-KZG, Plonk-KZG
  - **Pros**: Universal trusted setup
  - **Cons**: Proofs are **larger** than Groth16, P is **slower** than Groth16, also not post-quantum
  - Counterpoint for P can use more flexible intermediate representations than circuits and R1CS

Thank you!!