Zero Knowledge Proofs

Polynomial Commitments based on Pairing and Discrete Logarithm

Instructors: Dan Boneh, Shafi Goldwasser, Dawn Song, Justin Thaler, **Yupeng Zhang**





Stanford University

Berkeley UNIVERSITY OF CALIFORNIA

GEORGETOWN UNIVERSITY

TEXAS A&M UNIVERSITY

Recall: how to build an efficient SNARK?

A polynomial commitment scheme

A polynomial interactive oracle proof (IOP)

SNARK for general circuits

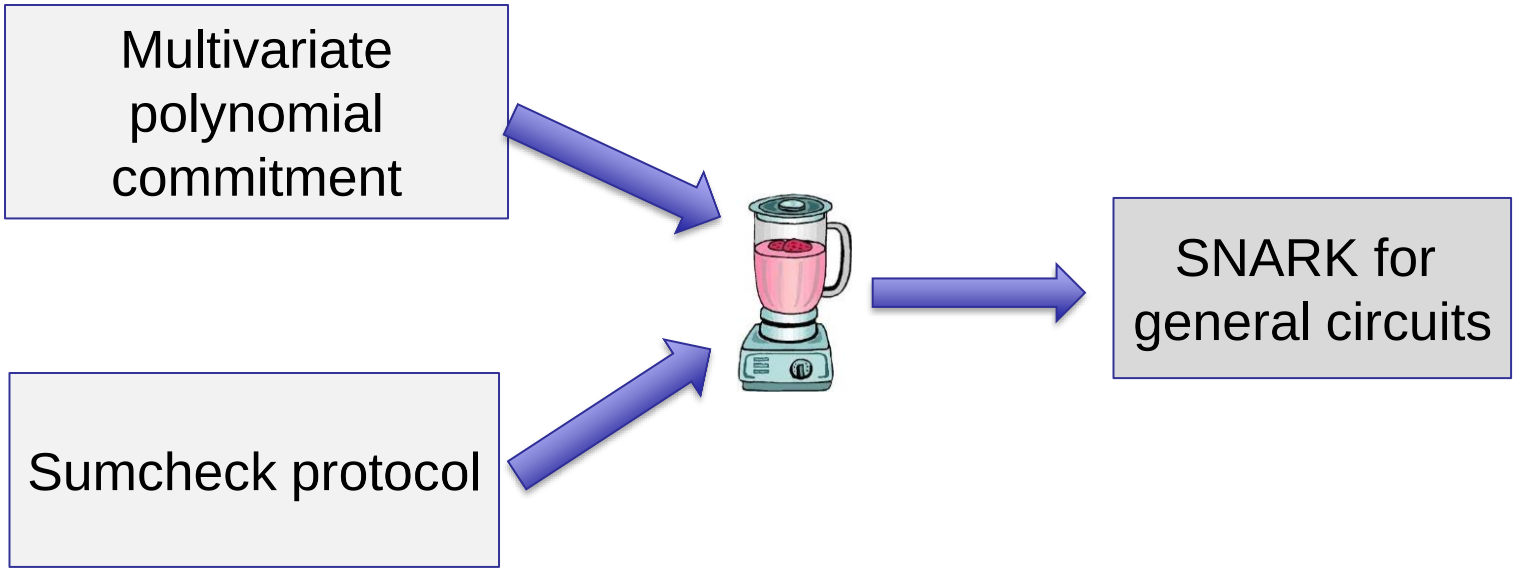
Recall: Plonk

Univariate polynomial commitment

Plonk Polynomial IOP

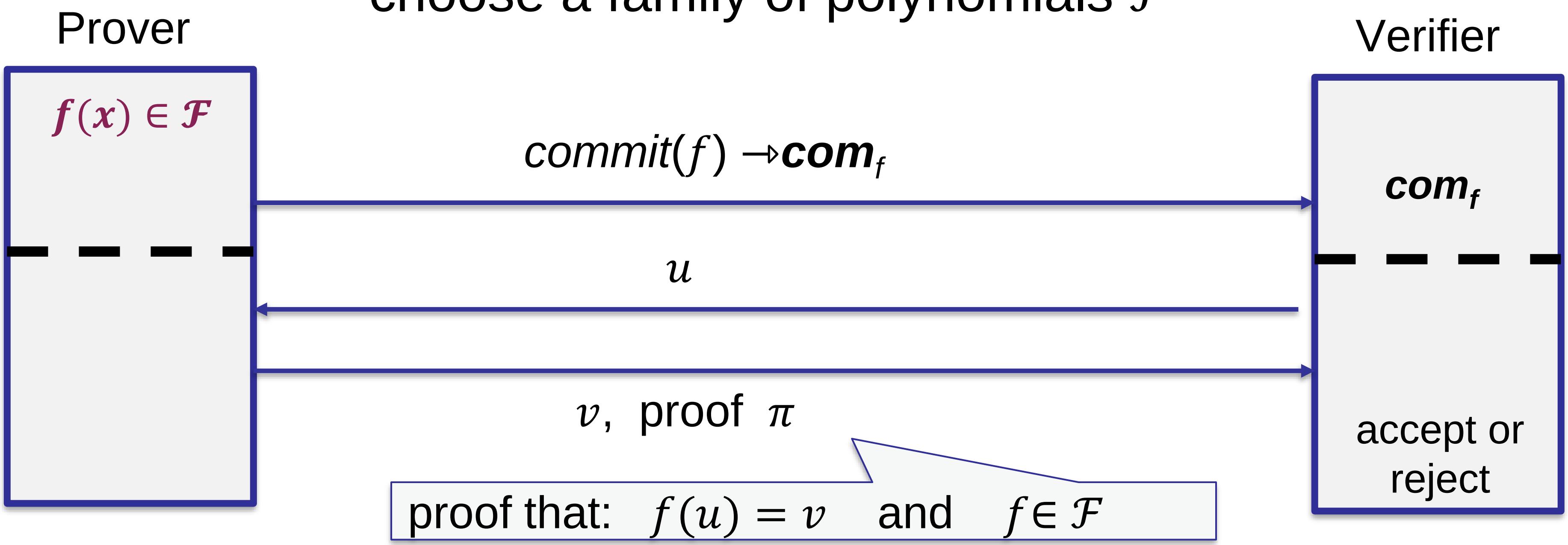
SNARK for general circuits

Recall: interactive proofs



What is a polynomial commitment

choose a family of polynomials $\mathcal{F}$



Definitions of polynomial commitments

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

Knowledge sound: for every poly. time adversary $A = (A_0, A_1)$ such that

$$keygen(\lambda, \mathcal{F}) \rightarrow gp,\ A_0(gp) \rightarrow com_f,\ A_1(gp, u) \rightarrow v, \pi:$$
$$\Pr[\ V(vp, x, \pi) = \text{accept}\ ] = 1$$

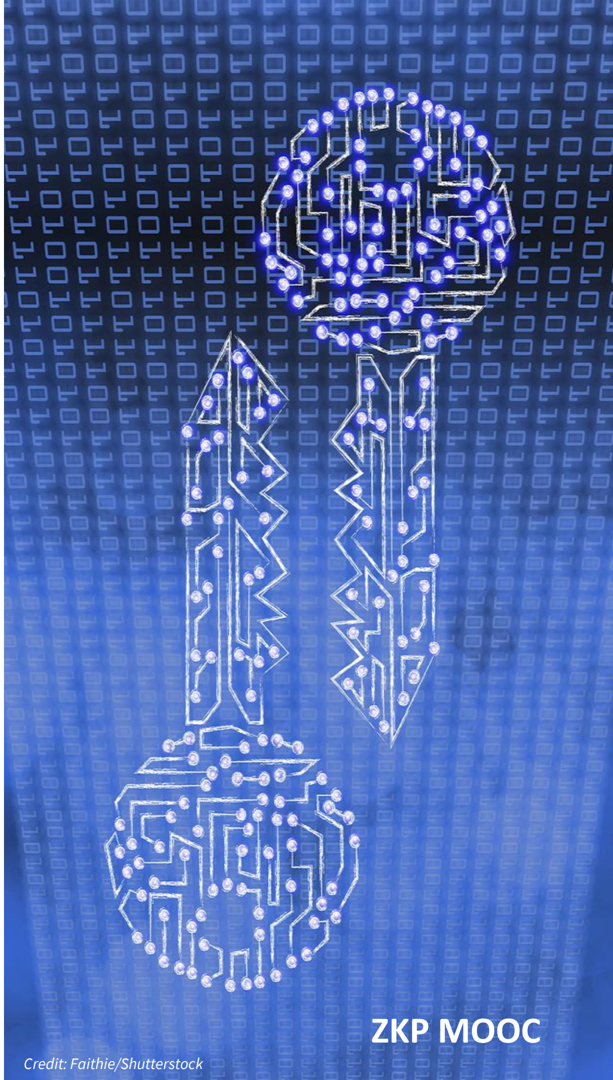
there is an efficient **extractor** E (that uses A) s.t.

$$keygen(\lambda, \mathcal{F}) \rightarrow gp,\ A_0(gp) \rightarrow com_f,\quad \boldsymbol{E(gp, com_f) \rightarrow f}:$$
$$\Pr[f(u) = v \text{ and } f(x) \in \mathcal{F}] > 1 - \epsilon \text{ (for a negligible } \epsilon)$$

Plan of this lecture

- Background
- KZG polynomial commitment and its variants
- Bulletproofs and other schemes based on discrete-log

Background



Credit: Faithie/Shutterstock

Group

A set $\mathbb{G}$ and an operation *

1. Closure: For all $a, b \in \mathbb{G}$, $a * b \in \mathbb{G}$
2. Associativity: For all $a, b, c \in \mathbb{G}$, $(a * b) * c = a * (b * c)$
3. Identity: There exists a unique element $e \in \mathbb{G}$ s.t. for every $a \in \mathbb{G}$, $e * a = a * e = a$.
4. Inverse: For each $a \in \mathbb{G}$, there exists $b \in \mathbb{G}$ s.t. $a * b = b * a = e$

E.g.: integers $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$ under add +
positive integers mod prime p : $\{1,2,\ldots,p-1\}$ under mult ×
elliptic curves

Generator of a group

- An element g that generates all elements in the group by taking all powers of g

Examples: $\mathbb{Z}_7^* = \{1,2,3,4,5,6\}$

$$3^1 = 3; \quad 3^2 = 2; \quad 3^3 = 6;$$
$$3^4 = 4; \quad 3^5 = 5; \quad 3^6 = 1; \quad \bmod 7$$

Discrete logarithm assumption

- A group $\mathbb{G}$ has an alternative representation as the powers of the generator g: $\{g, g^2, g^3, \ldots, g^{p-1}\}$
- Discrete logarithm problem:

  given $y \in \mathbb{G}$, $\quad \text{find } x \quad \text{s.t.} \quad g^x = y$
- Example: Find x such that $3^x = 4 \bmod 7$

- Discrete-log assumption: discrete-log problem is computationally hard

Diffie-Hellman assumption

- Computational DH assumption:

  Given $\mathbb{G}$, g, g^x, g^y, cannot compute g^{xy}

Bilinear pairing

- $(p, \mathbb{G}, g, \mathbb{G}_T, e)$
  - $\mathbb{G}$ and $\mathbb{G}_T$ are both multiplicative cyclic group of order p, g is the generator of $\mathbb{G}$.

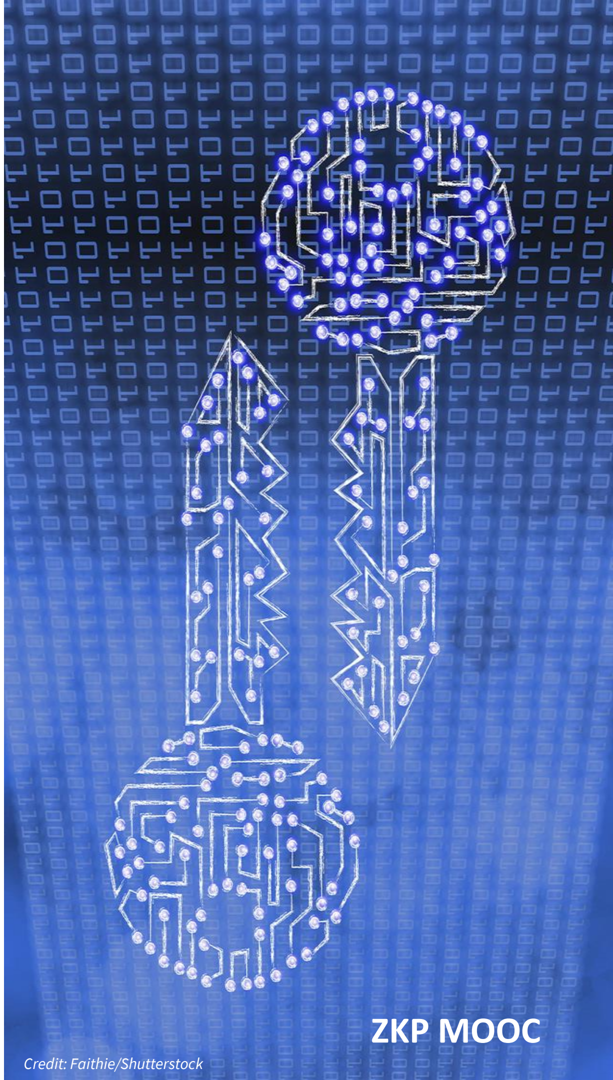
  $\mathbb{G}$:base group, $\mathbb{G}_T$ target group
- Pairing: $e(P^x, Q^y) = e(P, Q)^{xy} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$
- Example: $e(g^x, g^y) = e(g, g)^{xy} = e(g^{xy}, g)$

Given g^x and g^y, a pairing can check that some element $h = g^{xy}$ without knowing x and y

Example: BLS signature [Boneh–Lynn–Shacham'2001]

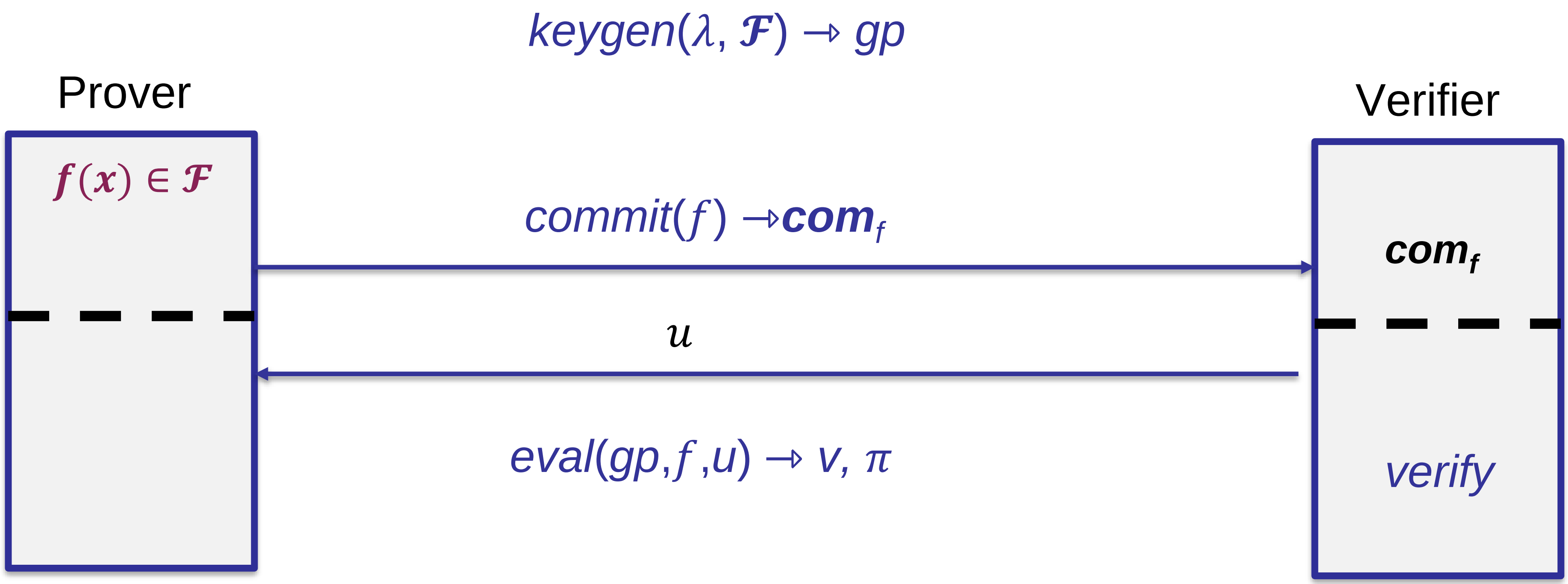
- Keygen: $p, \mathbb{G}, g, \mathbb{G}_T, e$
  private key x, public key g^x
- Sign(*sk*,*m*): $H(m)^x$, where H is a cryptographic hash that maps the message space to $\mathbb{G}$
- Verify(σ,*m*): $e(H(m), g^x) = e(\sigma, g)$

KZG polynomial commitment



The KZG poly-commit scheme (Kate-Zaverucha-Goldberg'2010)

Polynomial commitment



The KZG poly-commit scheme (Kate-Zaverucha-Goldberg'2010)

Bilinear Group $p, \mathbb{G}, g, \mathbb{G}_T, e$

Univariate polynomials $\boldsymbol{\mathcal{F}} = \mathbb{F}_p^{(\leq d)}[X]$

keygen$(\lambda, \boldsymbol{\mathcal{F}}) \rightarrow$ *gp:*

- Sample random $\tau \in \mathbb{F}_p$
- *gp* $= (g, g^{\tau}, g^{\tau^2}, \ldots, g^{\tau^d})$
- delete τ !! (trusted setup)

The KZG poly-commit scheme (Kate-Zaverucha-Goldberg'2010)

$gp = \left(g, g^{\tau}, g^{\tau^2}, \dots, g^{\tau^d}\right)$

$commit(gp, f) \to com_f$:

- $f(x) = f_0 + f_1 x + f_2 x^2 + \cdots + f_d x^d$
- $com_f = g^{f(\tau)}$

$$= g^{f_0 + f_1\tau + f_2\tau^2 + \cdots + f_d\tau^d}$$

$$= (g)^{f_0} \cdot (g^{\tau})^{f_1} \cdot \left(g^{\tau^2}\right)^{f_2} \cdot \ldots \cdot \left(g^{\tau^d}\right)^{f_d}$$

The KZG poly-commit scheme (Kate-Zaverucha-Goldberg'2010)

$gp = (g, g^{\tau}, g^{\tau^2}, \ldots, g^{\tau^d})$

$eval(gp, f, u) \to v, \pi$:

- $f(x) - f(u) = (x-u)q(x)$, as u is a root of $f(x) - f(u)$
- Compute $q(x)$ and $\pi = g^{q(\tau)}$, using gp

The KZG poly-commit scheme (Kate-Zaverucha-Goldberg'2010)

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

Honest prover: $com_f = g^{f(\tau)}$, $\pi = g^{q(\tau)}$, $v = f(u)$

verify(*gp*, com_f, *u*, *v*, π) :

- Idea: check the equation at point τ: $\cancel{g^{f(\tau)-f(u)} = g^{(\tau-u)q(\tau)}}$
- Challenge: only know $g^{\tau-u}$ and $g^{q(\tau)}$
- Solution: pairing! $e(com_f / g^{v}, g) = e(g^{\tau-u}, \pi)$

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

KZG polynomial commitment

Univariate polynomials of degree <= *d*

Prover

$gp = (g, g^{\tau}, g^{\tau^2}, \dots, g^{\tau^d})$

Verifier

$\boldsymbol{f(x)}$

$com_f = g^{f(\tau)}$

com$_f$

u

v, proof $\pi = g^{q(\tau)}$

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

$e(com_f / g^v, g) = e(g^{\tau - u}, \pi)$

Soundness of the KZG scheme

q-Strong Bilinear Diffie-Hellman (q-SBDH) assumption:

Given $(p, \mathbb{G}, g, \mathbb{G}_T, e)$, $(g, g^{\tau}, g^{\tau^2}, \ldots, g^{\tau^d})$, cannot compute $e(g,g)^{\frac{1}{\tau - u}}$ for any u

Soundness of the KZG scheme

Proof by contradiction: Suppose $v^* \neq f(u)$, π^* pass the verification

- $e(com_f / g^{v^*}, g) = e(g^{\tau - u}, \pi^*)$
- $e(g^{f(\tau) - v^*}, g) = e(g^{\tau - u}, \pi^*)$ Knowledge assumption later

$\Leftrightarrow e(g^{\boldsymbol{f(\tau) - f(u) + f(u) - v^*}}, g) = e(g^{\tau - u}, \pi^*)$, define $\delta = f(u) - v^*$

$\Leftrightarrow e(g^{\boldsymbol{(\tau - u) q(\tau) + \delta}}, g) = e(g^{\tau - u}, \pi^*)$

$\Leftrightarrow e(g, g)^{(\tau - u) q(\tau) + \delta} = e(g, \pi^*)^{\tau - u}$

$\Leftrightarrow e(g, g)^{\delta} = \left(e(g, \pi^*) / e(g, g)^{q(\tau)} \right)^{\tau - u}$

$\Leftrightarrow e(g, g)^{\frac{\delta}{\tau - u}} = e(g, \pi^*) / e(g, g)^{q(\tau)}$ breaks q-SBDH assumption!

Knowledge soundness and KoE assumption

- Why the prover knows f s.t. $com_f = g^{f(\tau)}$

Knowledge of exponent assumption:

- $g, g^{\tau}, g^{\tau^2}, \dots, g^{\tau^d}$
- Sample random α, compute $g^{\alpha}, g^{\alpha\tau}, g^{\alpha\tau^2}, \dots, g^{\alpha\tau^d}$
- $com_f = g^{f(\tau)}$, $com_f' = g^{\alpha f(\tau)}$
- If $e\left(com_f, g^{\alpha}\right) = e(com_f', g)$, there exists an extractor E that extracts f s.t. $com_f = g^{f(\tau)}$

KZG with knowledge soundness

- Keygen: gp includes both $g, g^{\tau}, g^{\tau^2}, \dots, g^{\tau^d}$ and $g^{\alpha}, g^{\alpha\tau}, g^{\alpha\tau^2}, \dots, g^{\alpha\tau^d}$
- Commit: $com_f = g^{f(\tau)}, com'_f = g^{\alpha f(\tau)}$
- Verify: additionally checks $e\left(com_f, g^{\alpha}\right) = e(com'_f, g)$

- Knowledge soundness proof: extract f in the first step by the KoE assumption

Generic group model (GGM) [Shoup'97, Maurer'05]

- (Informal) Adversary is only give an oracle to compute the group operation.

  E.g., given $g, g^{\tau}, g^{\tau^2}, \dots, g^{\tau^d}$, Adv can only compute their linear combinations.

- GGM can replace the KoE assumption and reduce the commitment size in KZG.

See book "A Graduate Course in Applied Cryptography" by Dan Boneh and Victor Shoup, section 16.3 for more details

KZG polynomial commitment

Univariate polynomials of degree <= d

$gp = (g, g^{\tau}, g^{\tau^2}, \ldots, g^{\tau^d})$

Prover

$\boldsymbol{f(x)}$

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

$com_f = g^{f(\tau)}$

u

v, proof $\pi = g^{q(\tau)}$

Verifier

$\boldsymbol{com_f}$

$e(com_f / g^v, g) = e(g^{\tau - u}, \pi)$

Properties of the KZG poly-commit

- Keygen: trusted setup!
- Commit: O(d) group exponentiations, O(1) commitment size
- Eval: O(d) group exponentiations

  $q(x)$ can be computed efficiently in linear time!
- Proof size: O(1), 1 group element
- Verifier time: O(1), 1 pairing

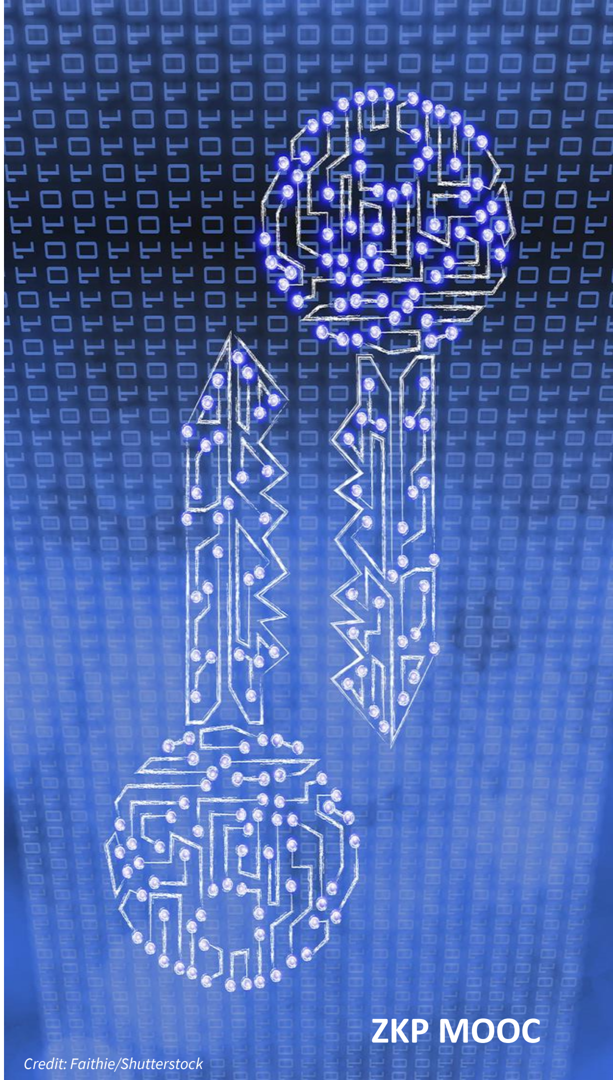
Ceremony

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

- $gp = \left(g^{\tau}, g^{\tau^2}, \ldots, g^{\tau^d}\right) = (g_1, g_2, \ldots, g_d)$
- Sample random *s*, update $gp' = (g_1', g_2', \ldots, g_d') = (g_1^s, g_2^{s^2}, \ldots, g_d^{s^d})$
  $= \left(g^{\tau s}, g^{(\tau s)^2}, \ldots, g^{(\tau s)^d}\right)$ with secret $\tau \cdot s$!
- Check the correctness of *gp'*
  1. The contributor knows s s.t. $g_1' = (g_1)^s$
  2. gp' consists of consecutive powers $e(g_i', g_1') = e(g_{i+1}', g)$, and $g_1' \neq 1$

  See [Nikolaenko-Ragsdale-Bonneau-Boneh'22]

Variants of KZG polynomial commitment



Multivariate poly-commit [Papamanthou-Shi-Tamassia'13]

E.g., $f(x_1, \dots, x_k) = x_1 x_3 + x_1 x_4 x_5 + x_7$

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

- *Keygen*: sample $\tau_1, \tau_2, \dots, \tau_k$, compute *gp* as *g* raised to all possible monomials of $\tau_1, \tau_2, \dots, \tau_k$ e.g., 2^k monomials for multilinear polynomial
- *Commit*: $com_f = g^{f(\tau_1, \tau_2, \dots, \tau_k)}$
- *Eval*: compute $\pi_i = g^{q_i(\vec{\tau})}$
- Verify: $e\left(com_f / g^v, g\right) = \prod_{i=1}^{k} e(g^{\tau_i - u_i}, \pi_i)$

$O(\log N)$ proof size and verifier time

Achieving zero-knowledge [ZGKPP'2018]

- See lecture 1 for the formal definition
- Plain KZG is not ZK. E.g., $com_f = g^{f(\tau)}$ is deterministic

Solution: masking with randomizers

- Commit: $com_f = g^{f(\tau)+r\eta}$
- Eval: $f(x) + ry - f(u) = (x-u)(q(x) + r'y) + y(r - r'(x-u))$

$$\pi = g^{q(\tau)+r'\eta}, g^{r-r'(\tau-u)}$$

Batch opening: single polynomial

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

Key idea:

- Extrapolate $f(u_1), \dots, f(u_m)$ to get $h(x)$
- $f(x) - h(x) = \prod_{i=1}^{m}(x - u_i)\, q(x)$
- $\pi = g^{q(\tau)}$
- $e\left(com_f / g^{h(\tau)}, g\right) = e\left(g^{\prod_{i=1}^{m}(\tau - u_i)}, \pi\right)$

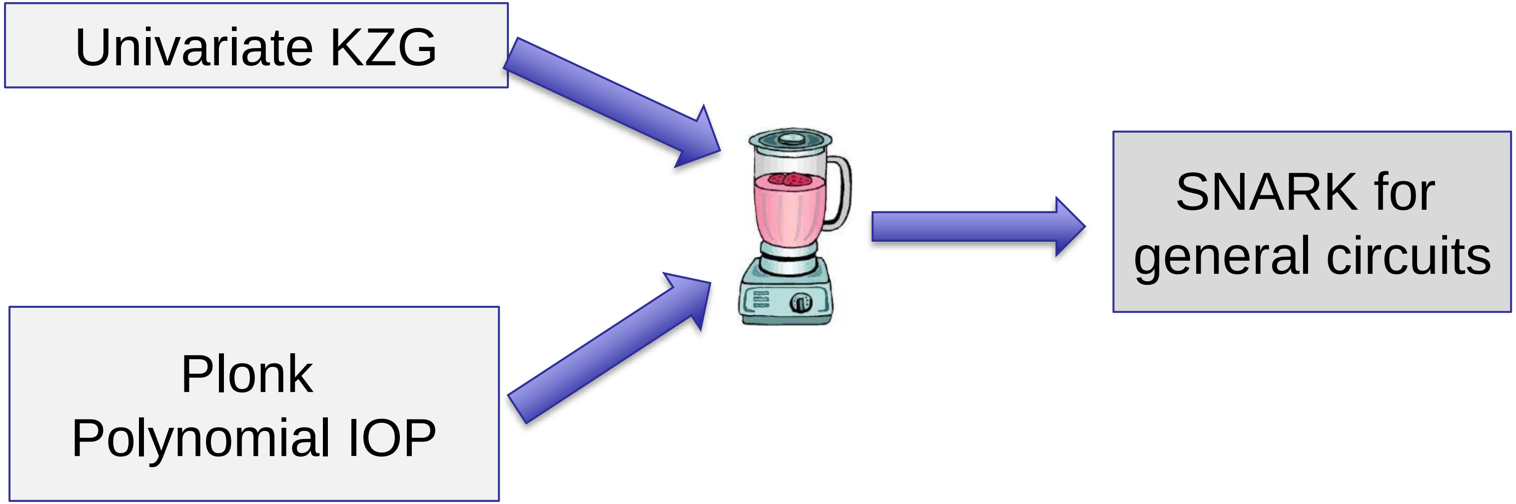
Batch opening: multiple polynomials

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

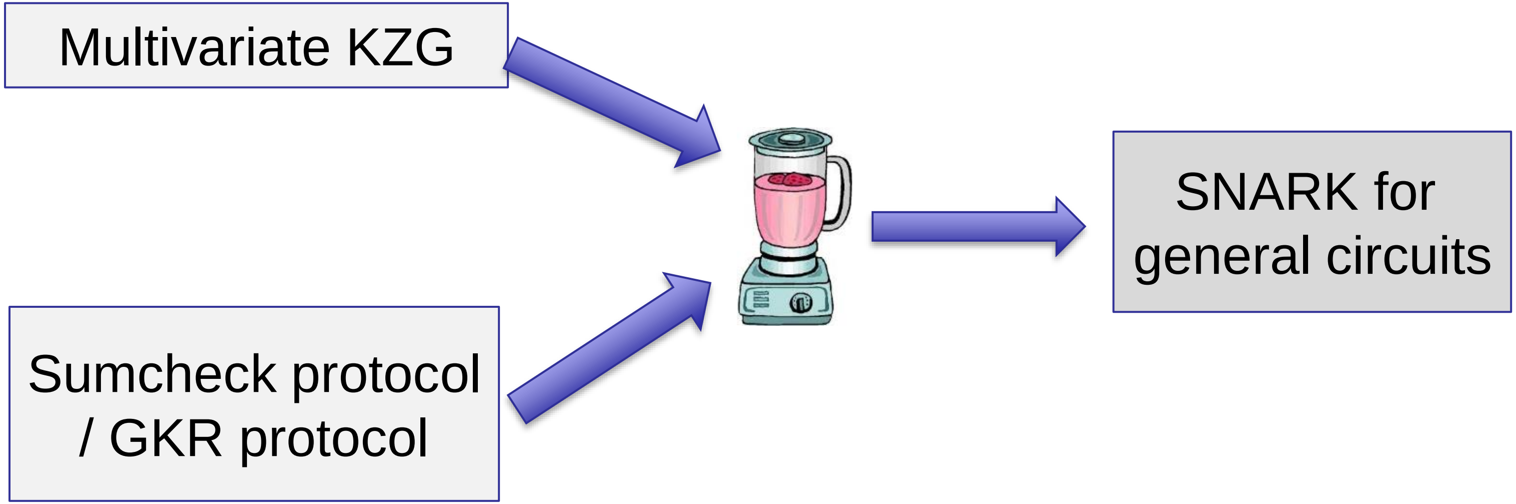
Key idea:

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

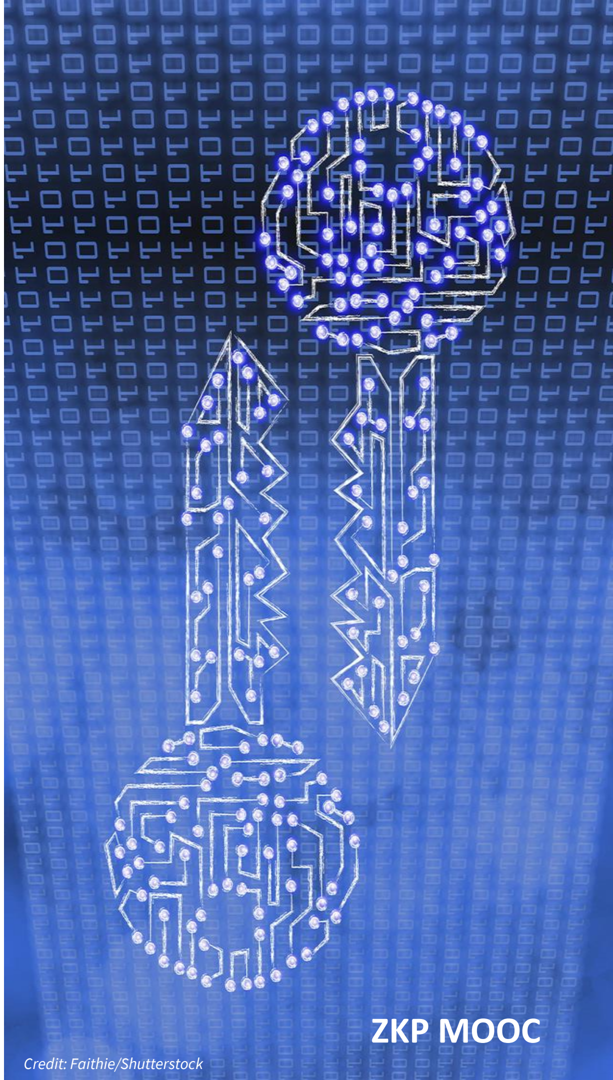
Plonk [Gabizon-Williamson-Ciobotaru’20]



vSQL[ZGKPP'17], Libra[XZZPS'19]



Polynomial commitments based on discrete-log



Credit: Faithie/Shutterstock

Recall: Pros and Cons of the KZG poly-commit

- ✓ Commitment and proof size: O(1), 1 group element
- ✓ Verifier time: O(1) pairing

- ✗ Keygen: trusted setup

Bulletproofs [BCCGP'16, BBBPWM'18]

Transparent setup: sample random $gp = (g_0, g_1, g_2, \ldots, g_d)$ in $\mathbb{G}$

Commit: $f(x) = f_0 + f_1 x + f_2 x^2 + \cdots + f_d x^d$

$com_f = g_0^{f_0} g_1^{f_1} g_2^{f_2} \ldots g_d^{f_d}$ Pedersen vector commitment

High-level idea

Prover

Verifier

f_0, f_1, f_2, f_3

f_0', f_1'

$$com_f = g_0^{f_0} g_1^{f_1} g_2^{f_2} g_3^{f_3}$$

$$u$$

$$v = f_0 + f_1 u + f_2 u^2 + f_3 u^3$$

$$com_{f'} = g_0'^{f_0'} g_1'^{f_1'}$$

$$v' = f_0' + f_1' u$$

com_f

$com_{f'}$

Poly-commitment based on Bulletproofs

$gp = (g_0, g_1, g_2, g_3)$

Prover

f_0, f_1, f_2, f_3

$rf_0 + f_2, rf_1 + f_3$

Verifier

$com_f = g_0^{f_0} g_1^{f_1} g_2^{f_2} g_3^{f_3}$

$v = f_0 + f_1 u + f_2 u^2 + f_3 u^3$

$L = g_2^{f_0} g_3^{f_1} \quad R = g_0^{f_2} g_1^{f_3}$

$v_L = f_0 + f_1 u \quad v_R = f_2 + f_3 u$

$v = v_L + v_R u^2$

r

$com' = L^r \cdot com_f \cdot R^{r^{-1}}$

$gp' = (g_0^{r^{-1}} g_2, g_1^{r^{-1}} g_3)$

$v' = rv_L + v_R$

Poly-commitment based on Bulletproofs

$$com_f = g_0^{f_0} g_1^{f_1} g_2^{f_2} g_3^{f_3}, \quad L = g_2^{f_0} g_3^{f_1} \qquad R = g_0^{f_2} g_1^{f_3}$$

$$com' = L^r \cdot com_f \cdot R^{r^{-1}} = g_0^{f_0 + r^{-1} f_2} g_2^{r f_0 + f_2} \cdot g_1^{f_1 + r^{-1} f_3} g_3^{r f_1 + f_3}$$

$$= \left(g_0^{r^{-1}} g_2\right)^{r f_0 + f_2} \cdot \left(g_1^{r^{-1}} g_3\right)^{r f_1 + f_3}$$

$$gp' = \left(g_0^{r^{-1}} g_2, g_1^{r^{-1}} g_3\right)$$

Poly-commitment based on Bulletproofs

$gp = (g_0, g_1, g_2, g_3)$

Prover

f_0, f_1, f_2, f_3

$rf_0 + f_2, rf_1 + f_3$

Verifier

$com_f = g_0^{f_0} g_1^{f_1} g_2^{f_2} g_3^{f_3}$

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

Prover → Verifier: $L = g_2^{f_0} g_3^{f_1}$ $\quad R = g_0^{f_2} g_1^{f_3}$

$v_L = f_0 + f_1 u$ $\quad v_R = f_2 + f_3 u$

Verifier: $v = v_L + v_R u^2$

Verifier → Prover: r

$com' = L^r \cdot com_f \cdot R^{r^{-1}}$

$gp' = (g_0^{r^{-1}} g_2, g_1^{r^{-1}} g_3)$

$v' = rv_L + v_R$

Poly-commitment based on Bulletproofs

- Eval
  1. Compute L, R, v_L, v_R
  2. Receive r from verifier, reduce f to f' of degree $\frac{d}{2}$
  3. Update the bases gp'
- Verify
  1. Check $v = v_L + v_R u^{d/2}$
  2. Generate r randomly
  3. Update $com' = L^r \cdot com_f \cdot R^{r^{-1}}, gp', v' = rv_L + v_R$

Recurse $\log d$ times

Properties of Bulletproofs

- Keygen: $O(d)$, transparent setup!
- Commit: $O(d)$ group exponentiations, $O(1)$ commitment size
- Eval: $O(d)$ group exponentiations

  (non-interactive via Fiat Shamir)
- Proof size: $O(\log d)$
- Verifier time: $O(d)$

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

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

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

Dory [Lee'2021]

- Improving verifier time to $O(\log d)$
- Key idea: delegating the structured verifier computation to the prover using inner pairing product arguments [BMMTV'2021]
- Also improves the prover time to $O(\sqrt{d})$ exponentiations plus $O(d)$ field operations

Dark [Bünz-Fisch-Szepieniec'20]

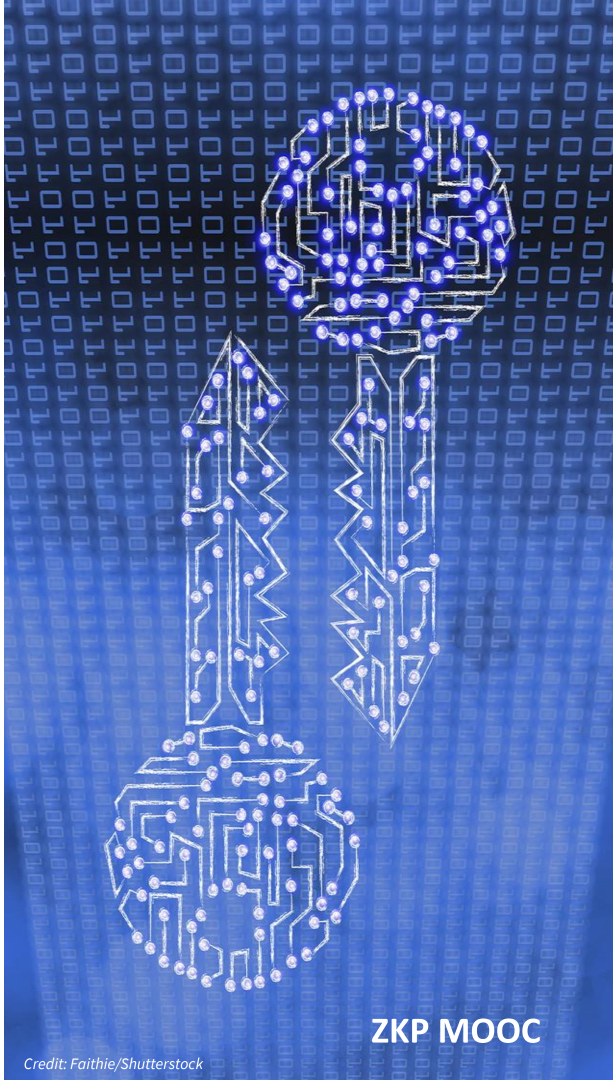
- Achieves $O(\log d)$ proof size and verifier time

- Group of unknown order

Summary

Scheme	Prover	Proof size	Verifier	Trusted Setup	Crypto primitive
KZG	$O(d)$	$O(1)$	$O(1)$	✓	pairing
Bullet-proofs	$O(d)$	$O(\log d)$	$O(d)$	×	discrete-log
Hyrax	$O(d)$	$O(\sqrt{d})$	$O(\sqrt{d})$	×	discrete-log
Dory	$O(d)$	$O(\log d)$	$O(\log d)$	×	pairing
Dark	$O(d)$	$O(\log d)$	$O(\log d)$	×	unknown order group

End of Lecture



Credit: Faithie/Shutterstock